

توضيح

كويك يسك

مدى

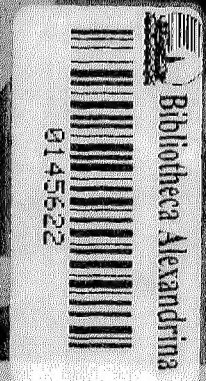
ترجمة ومراجعة

د.م. سرور على ابراهيم سرور

*uick
Basic*

تقديم

أ.د. عبد الله بن عبد الله العبيد



المكتبة الأكاديمية

لوزيڻ
كويڪ بيسڪ

توضيح كويك يسك

تأليف

هاري ومورات تانيك واوڤو بوش

ترجمة ومراجعة

د. م. سرور علي ابراهيم سرور

الاستاذ المشارك بقسم الأساليب الكمية

كلية الاقتصاد والادارة

جامعة الملك سعود - فرع القصيم

تقديم

د. محمد الله بن محمد الله الحبيد

عميد كلية الاقتصاد والادارة

جامعة الملك سعود - فرع القصيم



الناشر

المكتبة الأكاديمية

١٩٩٤

بسم الله الرحمن الرحيم

تقديم

الحمد لله رب العالمين، والصلاة والسلام على أشرف المرسلين سيدنا محمد وعلى آله وصحبه ومن تبعهم بإحسان إلى يوم الدين.

ويعد...

فهذا كتاب جديد عن إحدى لغات الكمبيوتر التي بدأ استخدامها منذ خمسة وعشرين عاماً ولكن شتان بين الصيغة الأولى لهذه اللغة والصيغة المعروضة حالياً في هذا الكتاب، فقد ظهرت لغة البيسك كلفة مبسطة لتعليم المبتدئين كتابة برامج للكمبيوتر. وقد كانت امكانيات هذه اللغة مبسطة لدرجة جعلت من غير المتوقع أن تأخذ هذه اللغة شكلها الحالي. وكما هو الشأن في مجال الكمبيوتر دائماً فلم تأخذ لغة البيسك شكلها الحالي فجأة بل حدث هذا من خلال التطور المستمر لهذه اللغة والذي جعل البعض في بداية الثمانينيات الميلادية يعتقد أنه من المستحيل إعداد صيغة نمطية لهذه اللغة لا شيء إلا لكثرة الصيغ التجارية المتوفرة لها في الأسواق، إلا أن البعض اقترح أن تكون صيغة بييسك أجهزة الميكروكمبيوتر من طراز IBM (IBM PC BASIC) وصيغة البييسك المسماة GW-BASIC من شركة ميكروسوفت كصيغ نمطية للغة البييسك، إلا أن صيغ بييسك السريع المقدمة من شركة ميكروسوفت وكذلك صيغ تربو بييسك المقدمة من شركة بورلاند العالمية لم تغير فقط من محتويات لغة البييسك بل أنها أدخلتها كذلك في مصاف اللغات المهمة المتوقع الاعتماد عليها بشدة في المستقبل. وأحد الأسباب الرئيسية لذلك هو أن هذه الصيغ من لغة البييسك هي صيغ مرتبة structured تصلح للبرمجة المرتبة.

ولم يقع اختيار الكلية على هذا الكتاب لترجمته وتقديمه للقارئ العربي لحداثة المادة العلمية المقدمة فيه فحسب بل لسهولة عرضها وترتيب مواضيعها. وعلى الرغم من أن الكتاب مقدم للقارئ المبتدئ وكذلك للمبرمجين إلا أنه لا يحتاج لشيء في دراسته سوى الإلمام باستخدام لوحة مفاتيح الكمبيوتر ونظام تشغيله. وهذه متطلبات ميسورة لمن يرغب في تعلم هذه اللغة المهمة.

ولا يفوتنا في هذه السطور أن ننوه بالجهد العلمي المشكور الذي قام به الدكتور/ سرور على ابراهيم سرور الاستاذ المشارك بقسم الأساليب الكمية بالكلية، حيث أخذ على عاتقه مسئولية

الكتاب كاملة في الترجمة ومراجعة الترجمة أيضاً، فجزاه الله خير الجزاء على هذا العمل العلمي
في ميدان من أهم ميادين العلم في حياتنا المعاصرة.

نسأل الله تعالى أن يجعل كل أعمالنا خاصة لوجهه، إنه سميع مجيب.

د. محمد الله بن محمد الله الحبيد

عميد كلية الاقتصاد والإدارة

جامعة الملك سعود - فرع القصيم

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

« قل لو كان البحر مداداً لكلمات ربي لنفد البحر
قبل أن تنفذ كلمات ربي ولو جئنا بمثله مدداً ».

صدق الله العظيم

مقدمة المترجم

بسم الله الرحمن الرحيم

الحمد لله، نحمده ونستعينه، ونصلي ونسلم على خير انبيائه، وخاتم رسله.

وبعد

لقد ظهرت لغة البيسك منذ خمسة وعشرين عاماً كلفة بسيطة لتعليم المبتدئين أساسيات البرمجة باستخدام الكمبيوتر. وشهدت هذه اللغة البسيطة تطورات هائلة على مدار هذه السنين جعلتها في مصاف لغات البرمجة الأساسية حالياً والمتوقع الإكثار من الاعتماد عليها مستقبلياً بعد أن أصبحت تدعم البرمجة المربطة والتي كانت تقتصر إليها الصيغ السابقة لهذه اللغة. ويرجع الفضل في جعل هذه اللغة قادرة على دعم البرمجة المربطة إلى صيغ كوك بييسك أو بييسك السريع المقدمة من شركة ميكروسوفت وإلى صيغ تربو بييسك المقدمة من شركة بورلاند العالمية. ويقدم هذا الكتاب، على هيئة دروس عملية مبسطة، الصيغة رقم 4.0 من لغة بييسك السريع، وسوف يستخدم اسم بييسك السريع في بقية الكتاب.

وفيما يلي بعض مميزات صيغ بييسك السريع عن صيغ بييسك السابقة لها وبصفة خاصة صيغ IBM PC BASIC و GW-BASIC والتي يعتبرها البعض صيغاً نمطية للغة البييسك :

١ - إمكانية كتابة الأسطر دون وضع أرقام لها، وإمكانية استخدام الفاصلة أو REM في كتابة الملاحظات، واستخدام اشباه الأوامر، واستخدام منظومات استاتيكية ومنظومات ديناميكية.

٢ - استخدام نوع جديد من أنواع المتغيرات العددية وهو الرقم الصحيح الطويل long integer وكذلك استخدام سلاسل وسجلات ثابتة الطول، والعمل من خلال قوائم وإمكانية الحصول على مساعدة أثناء العمل.

٣ - التوسع في مكونات التحكم التي سبق استخدامها في صيغ البييسك السابقة لها بإدخال مجموعات IF ومجموعات SELECT CASE ودورات DO.

٤ - استخدام البرامج الفرعية والتي تدعم البرمجة المربطة، وتكمن قوة البرامج الفرعية في إمكانية تمرير قيم إلى البرامج الفرعية مع بقاء متغيرات البرنامج الفرعي محلية له، كما ينطبق نفس المفهوم على النوال كذلك.

٥ - استخدام الملفات الثنائية، واستخدام مترجمات compilers للغة لترجمة البرامج ويمكن تنفيذ هذه البرامج اما من بيئة اللغة واما من بيئة نظام التشغيل DOS مباشرة.

٦ - استخدام المقاطع التي يمكن معالجتها كأجزاء مستقلة، واستخدام خاصية مشاركة المتغيرات عبر المقاطع المختلفة، وتجميع المقاطع المترجمة في مكتبة خاصة بها للاستعانة بها عند الحاجة لذلك، وإمكانية توصيل المقاطع المنفصلة مع بعضها البعض أو مع مقاطع المكتبة أو مع برامج بييسك سريع أخرى.

هذا وقد راعينا في ترجمة الكتاب الحفاظ على نفس الأسلوب والتنظيم المتبعين من قبل المؤلف، كما حافظنا كذلك على نفس التسلسل الذي قدمه المؤلف حيث تم سرد دروس الكتاب في تسلسلها طبقاً للحروف الأبجدية الانجليزية، كما تمت ترجمة تسلسل التعلم الذي يقترحه مؤلف الكتاب لدراسة الكتاب كدليل لمساعدة القارئ العربي المبتدئ في التعامل مع الكتاب.

ولا يسعنا في نهاية هذه المقدمة إلا أن نقدم خالص الشكر والعرفان إلى كل من ساهم في إخراج هذا الكتاب في هذه الصورة ونخص بالذكر سعادة الدكتور/ عبد الله بن عبد الله العبيد عميد كلية الاقتصاد والإدارة بجامعة الملك سعود فرع القصيم بعنيزة لتشجيعه الدائم على ترجمة أمهات الكتب إلى اللغة العربية وحثه المستمر لأعضاء هيئة التدريس بالكلية على الاستمرار في نفس هذا النهج، كما نشكر الاستاذ/ محمد رضوان لما ساهم به في انجاز هذا الكتاب، ولايفوتنا أن نشكر سعادة الاستاذ/ أحمد أمين محمود- مدير المكتبة الأكاديمية بالقاهرة- وسعادة المهندس/ حمدي قنديل، مدير الانتاج بالمكتبة الأكاديمية بالقاهرة، وجميع العاملين بالمكتبة لما يبذلونه من جهد صادق في اخراج مثل هذه الكتب العلمية القيمة للمكتبة العربية. ونرجو من الله أن نكون قد وفقنا لتقديم كتاب جيد وجديد للقارئ العربي.

والله ولي التوفيق.

المترجم

المحتويات

الدرس	الموضوع	صفحة
الأول	حول هذا الكتاب	٢٥
الثاني	عرض عام لبيسك السريع	٢٧
الثالث	عينة لجلسة مع بيسك السريع	٣١
الرابع	الترجمة من DOS	٣٧
الخامس	دالة القيمة المطلقة ABS	٤٠
السادس	دالة ASC	٤٢
السابع	دالة ATN	٤٥
الثامن	عبارة BEEP	٤٨
التاسع	عبارة BLOAD و BSAVE	٥٠
العاشر	عبارة CALL و CALLS	٥٤
الحادي عشر	عبارة CALL ABSOLUTE	٥٨
الثاني عشر	عبارة CALL INTERRUPT و CALL INT86OLD	٦٠
الثالث عشر	دوال CDBL و CINT و CLNG و CSNG	٦٥
الرابع عشر	عبارة CHAIN	٦٨
الخامس عشر	دالة CHR\$	٧٠
السادس عشر	عبارة CIRCLE	٧٢
السابع عشر	عبارة CLEAR	٧٤
الثامن عشر	عبارة CLOSE	٧٧
التاسع عشر	عبارة CLS	٧٩
العشرون	عبارة COLOR	٨٢
الواحد والعشرون	دالة COMMAND\$	٨٤
الثاني والعشرون	عبارة COMMON	٨٧
الثالث والعشرون	الثابت CONST	٩٣
الرابع والعشرون	دالة COS	٩٧

٩٩	دالة CSRLIN	الخامس والعشرون
١٠٢	نوال CVD و CVI و CVL و CVS	السادس والعشرون
١٠٥	عبارة DATA	السابع والعشرون
١٠٨	دالة وعبارة DATE\$	الثامن والعشرون
١١١	عبارة DECLARE	التاسع والعشرون
١١٧	عبارة DEF FN	الثلاثون
١٢١	عبارة DEF SEG	الواحد والثلاثون
١٢٤	عبارات DEFDBL و DEFINT و DEFLNG و DEFSTR	الثاني والثلاثون
١٢٧	عبارة البعد DIM	الثالث والثلاثون
١٣١	عبارة DO LOOP	الرابع والثلاثون
١٣٤	عبارة DRAW	الخامس والثلاثون
١٣٨	عبارة END	السادس والثلاثون
١٤٢	عبارة ENVIRON ودالة ENVIRON\$	السابع والثلاثون
١٤٤	دالة EOF	الثامن والثلاثون
١٤٧	عبارة ERASE	التاسع والثلاثون
١٥٠	دالتا ERDEV\$ و ERDEV	الأربعون
١٥٣	دالتا ERR و ERL وعبارة ON ERROR GOTO	الواحد والأربعون
١٥٨	عبارة ERROR	الثاني والأربعون
١٦١	عبارة EXIT	الثالث والأربعون
١٦٤	دالة EXP	الرابع والأربعون
١٦٦	عبارة FIELD	الخامس والأربعون
١٦٩	دالة FILEATTR	السادس والأربعون
١٧٢	عبارات FILES و CHDIR و MKDIR و RMDIR	السابع والأربعون
١٧٧	دالة FIX	الثامن والأربعون
١٧٩	عبارة FOR.. NEXT	التاسع والأربعون
١٨٣	دالة FRE	الخمسون
١٨٦	دالة FREEFILE	الواحد والخمسون
١٨٩	عبارة FUNCTION	الثاني والخمسون

١٩٢	عبارتا GET و PUT	الثالث والخمسون
١٩٦	عبارتا GET Graphics و PUT Graphics	الرابع والخمسون
٢٠٠	تكوين GOSUB.. RETUTRN	الخامس والخمسون
٢٠٥	عبارة GOTO	السادس والخمسون
٢٠٨	دالة HEX\$	السابع والخمسون
٢١٠	عبارة IF.. THEN.. ELSE	الثامن والخمسون
٢١٤	عبارة \$INCLUDE	التاسع والخمسون
٢١٧	دالة INKEY\$	الستون
٢٢٠	عبارات INP و OUT و WAIT	الواحد والستون
٢٢٢	عبارة INPUT	الثاني والستون
٢٢٧	عبارة INPUT#	الثالث والستون
٢٣٠	عبارة INPUT\$	الرابع والستون
٢٣٣	دالة INSTR	الخامس والستون
٢٣٦	دالة INT	السادس والستون
٢٣٨	عبارة IOCTL ودالة IOCTL\$	السابع والستون
٢٤٠	عبارات KEY	الثامن والستون
٢٤٦	عبارة KILL	التاسع والستون
٢٤٨	الاسماء LABLES	السبعون
٢٥٠	دالتا LBOUND و UBOUND	الواحد والسبعون
٢٥٣	دالتا LCASE\$ و UCASE\$	الثاني والسبعون
٢٥٦	دالة LEFT\$	الثالث والسبعون
٢٥٩	دالة LEN	الرابع والسبعون
٢٦٢	عبارة LET	الخامس والسبعون
٢٦٤	عبارة LINE	السادس والسبعون
٢٦٦	عبارتا LINE INPUT# و LINE INPUT	السابع والسبعون
٢٧٠	دالة LOC	الثامن والسبعون
٢٧٣	عبارة LOCATE	التاسع والسبعون
٢٧٦	عبارتا LOCK و UNLOCK	الثمانون

٢٧٨	دالة LOF	الواحد والثمانون
٢٨٠	عبارة LOG	الثاني والثمانون
٢٨٢	دالة LPOS	الثالث والثمانون
٢٨٤	عبارة LPRINT و LPRINT USING	الرابع والثمانون
٢٨٧	عبارة LSET و RSET	الخامس والثمانون
٢٩١	دالة LTRIM\$ و RTRIM\$	السادس والثمانون
٢٩٤	دالة وعبارة MID\$	السابع والثمانون
٢٩٨	دوال MKD\$ و MK1\$ و MKL\$ و MKS\$	الثامن والثمانون
٣٠١	دوال MKDMBF\$ و MKSMBF\$ و CVDMBF و CVSMBF	التاسع والثمانون
٣٠٤	عبارة NAME.. AS..	التسعون
٣٠٦	دالة OCT\$	الواحد والتسعون
٣٠٨	عبارة GOSUB ON event	الثاني والتسعون
٣١١	عبارة GOTO ON.. و GOSUB ON..	الثالث والتسعون
٣١٤	عبارة OPEN	الرابع والتسعون
٣٢٠	عبارة COM و OPEN COM	الخامس والتسعون
٣٢٤	عبارة OPTION BASE	السادس والتسعون
٣٢٧	عبارة PAINT	السابع والتسعون
٣٢٩	عبارة PALETTE USING و PALETTE	الثامن والتسعون
٣٣١	عبارة PCOPY	التاسع والتسعون
٣٣٣	دالة PEEK وعبارة POKE	المائة
٣٣٦	دالة PEN	المائة والواحد
٣٣٨	عبارات PEN ON و PEN OFF و PEN STOP	المائة والاثنين
٣٣٩	دالة وعبارة PLAY	المائة والثلاثة
٣٤٣	عبارات PLAY ON و PLAY OFF و PLAY STOP	المائة والأربعة
٣٤٥	دالة PMAP	المائة والخمسة
٣٤٧	دالة POINT	المائة والستة
٣٤٩	دالة POS	المائة والسبعة
٣٥١	عبارة PRESET	المائة والثمانية

٣٥٣	عبارة PRINT	المائة والتسعة
٣٥٧	عبارة PRINT USING	المائة والعشرة
٣٦١	عبارتا PRINT# و PRINT# USING	المائة والإحدى عشر
٣٦٤	عبارة PSET	المائة والإثنى عشر
٣٦٦	عبارة RANDOMIZE	المائة والثلاثة عشر
٣٦٩	عبارة READ	المائة والأربعة عشر
٣٧٣	عبارة REDIM	المائة والخمسة عشر
٣٧٦	عبارة REM	المائة والستة عشر
٣٧٨	عبارة RESET	المائة والسبعة عشر
٣٨٠	عبارة RESTORE	المائة والثمانية عشر
٣٨٣	عبارة RESUME	المائة والتسعة عشر
٣٨٦	دالة RIGHT\$	المائة والعشرون
٣٨٩	دالة RND	المائة والواحد والعشرون
٣٩٢	عبارة RUN	المائة والاثنين والعشرون
٣٩٥	دالة SADD	المائة والثلاثة والعشرون
٣٩٧	دالة وعبارة SCREEN	المائة والأربعة والعشرون
٤٠٣	دالة وعبارة SEEK	المائة والخمسة والعشرون
٤٠٦	عبارة SELECT CASE	المائة والستة والعشرون
٤١٠	دالة SYSTEM	المائة والسبعة والعشرون
٤١٢	دالة SGN	المائة والثمانية والعشرون
٤١٤	عبارة SHARED	المائة والتسعة والعشرون
٤١٧	عبارة SHELL	المائة وثلاثون
٤٢٠	دالة SIN	المائة والواحد والثلاثون
٤٢٢	عبارة SOUND	المائة والاثنين والثلاثون
٤٢٤	دالة SPACE\$	المائة والثلاثة والثلاثون
٤٢٦	دالة SPC	المائة والأربعة والثلاثون
٤٢٨	دالة SQR	المائة والخمسة والثلاثون
٤٣٠	عبارة STATIC	المائة والستة والثلاثون

٤٣٣	اشباه الأوامر \$DYNAMIC و \$STATIC	المائة والسبعة والثلاثون
٤٣٦	دالة STICK	المائة والثمانية والثلاثون
٤٣٧	عبارة STOP	المائة والتسعة والثلاثون
٤٣٩	دالة STR\$	المائة والأربعون
٤٤١	دالة STRIG	المائة والواحد والأربعون
	عبارات STRING ON و STRING OFF	المائة والاثنين وأربعون
٤٤٣	STRING STOP	
٤٤٥	دالة STRING\$	المائة والثلاثة والأربعون
٤٤٨	عبارتا SUB و END SUB	المائة والأربعة والأربعون
٤٥٢	عبارة SWAP	المائة والخمسة والأربعون
٤٥٥	عبارة SYSTEM	المائة والستة والأربعون
٤٥٧	دالة TAB	المائة والسبعة والأربعون
٤٥٩	دالة TAN	المائة والثمانية والأربعون
٤٦١	دالة وعبارة TIME\$	المائة والتسعة والأربعون
٤٦٤	دالة TIMER	المائة والخمسون
	عبارات TIMER ON و TIMER OFF و TIMER	المائة والواحد والخمسون
٤٦٦	STOP	
٤٦٩	عبارتا TRON و TROFF	المائة والاثنين والخمسون
٤٧٢	عبارتا TYPE و END TYPE	المائة والثلاثة والخمسون
٤٧٥	دالة VAL	المائة والأربعة والخمسون
٤٧٧	المتغيرات Variables	المائة والخمسة والخمسون
٤٨٢	دالتا VARPTR و VARSEG	المائة والستة والخمسون
٤٨٥	دالة VARPTR\$	المائة والسبعة والخمسون
٤٨٧	عبارة VIEW	المائة والثمانية والخمسون
٤٨٩	عبارة VIEW PRINT	المائة والتسعة والخمسون
٤٩١	عبارة WHILE.. WEND	المائة والستون
٤٩٤	عبارة WIDTH	المائة والواحد والستون
٤٩٧	عبارة WINDOW	المائة والاثنين والستون

٤٩٩	عبارتا WRITE و WRITE#	المائة والثلاثة والستون
٥٠٢	اصطلاحات وتعريفاتها	ملحق A
٥٠٦	استخدام منقح بيسك السريع	ملحق B
٥٢٧	جدول ASCII وكلمات بيسك السريع المحجوزة.	ملحق C
٥٢٩	مترجم وواصل سطر الأوامر	ملحق D
٥٣٤	المكتبات	ملحق E
٥٣٨	رسائل الخطأ	ملحق F
٥٥٢	تعاريف على بيسك السريع	ملحق G
٥٨١	قائمة بأهم المصطلحات	

تسلسل التعلم المقترح اتباعه

رقم الصفحة	الدرس	التسلسل
٢٥	الأول	١
٢٧	الثاني	٢
٥٠٦	ملحق B	٣
٣١	الثالث	٤
٣٧	الرابع	٥
٩٣	الثالث والعشرون	٦
٤٧٧	المائة والخامس والخمسون	٧
٤٧٢	المائة والثالث والخمسون	٨
٢٦٢	الخامس والسبعون	٩
٢٠٥	السادس والخمسون	١٠
٢٤٨	السبعون	١١
٢٧٣	التاسع والسبعون	١٢
٣٥٣	المائة وتسعة	١٣
٧٠	الخامس عشر	١٤
٤٤٥	المائة والثالث والأربعون	١٥
٤٢٤	المائة والثالث والثلاثون	١٦
١٧٩	التاسع والأربعون	١٧
٢١٠	الثامن والخمسون	١٨
٤٠٦	المائة والسادس والعشرون	١٩
٧٩	التاسع عشر	٢٠
٢١٠	الخامس والخمسون	٢١
١٣٨	السادس والثلاثون	٢٢
٤٥٧	المائة والسابع والأربعون	٢٣
٤٢٦	المائة والرابع والثلاثون	٢٤
٤٢	السادس	٢٥

٢٢٢	الثانى والستون	٢٦
٢٣٠	الرابع والسبعون	٢٧
٢٩٤	السابع والثمانون	٢٨
٣٨٦	المائة والعشرون	٢٩
٢٥٦	الثالث والسبعون	٣٠
٢٥٣	الثانى والسبعون	٣١
٢٩١	السادس والثمانون	٣٢
٢٣٣	الخامس والستون	٣٣
٣٦٩	المائة والاربعة عشر	٣٤
١٠٥	السابع والعشرون	٣٥
٤٦١	المائة والتاسع والاربعون	٣٦
١٠٨	الثامن والعشرون	٣٧
١٧٧	الثامن والاربعون	٣٨
٢٣٦	السادس والستون	٣٩
٤٠	الخامس	٤٠
٤٣٩	المائة والاربعون	٤١
٢٠٨	السابع والخمسون	٤٢
٣٠٦	الحادى والتسعون	٤٣
٤٥	السابع	٤٤
٩٧	الرابع والعشرون	٤٥
١٦٤	الرابع والاربعون	٤٦
٢٨٠	الثانى والثمانون	٤٧
٤٢٠	المائة والحادى والثلاثون	٤٨
٤٢٨	المائة والخامس والثلاثون	٤٩
٤٥٩	المائة والثامن والاربعون	٥٠
٤١٢	المائة والثامن والعشرون	٥١
٤٧٥	المائة والرابع والخمسون	٥٢
٤٥٢	المائة والخامس والاربعون	٥٣

٣٨٩	المائة والحادى والعشرون	٥٤
٣٦٦	المائة والثالث عشر	٥٥
٣٥٧	المائة وعشرة	٥٦
٣٧٦	المائة والسادس عشر	٥٧
٣٨٠	المائة والثامن عشر	٥٨
١٤٢	السابع والأربعون	٥٩
٣٠٤	التسعون	٦٠
٢٤٦	التاسع والستون	٦١
٤٩١	المائة والستون	٦٢
١٣١	الرابع والثلاثون	٦٣
٤٨	الثامن	٦٤
٤٢٢	المائة والثانى والثلاثون	٦٥
٣٣٩	المائة وثلاثة	٦٦
٣٤٣	المائة وأربعة	٦٧
٣٠٨	الثانى والتسعون	٦٨
٣١١	الثالث والتسعون	٦٩
١٥٣	الحادى والأربعون	٧٠
١٥٨	الثانى والأربعون	٧١
٣١٤	الرابع والتسعون	٧٢
١٦٦	الخامس والأربعون	٧٣
٢٨٧	الخامس والثمانون	٧٤
٣٦١	المائة والحادى عشر	٧٥
٢٢٧	الثالث والستون	٧٦
٢٣٠	الرابع والستون	٧٧
٢٦٦	السابع والسبعون	٧٨
٤٩٩	المائة والثالث والستون	٧٩
١٩٢	الثالث والخمسون	٨٠
١٤٤	الثامن والثلاثون	٨١

٢٧٠	الثامن والسبعون	٨٢
٤٠٣	المائة الخامس والعشرون	٨٣
٢٧٨	الحادي والثمانون	٨٤
١٨٦	الحادي والخمسون	٨٥
١٦٩	السادس والأربعون	٨٦
٧٧	الثامن عشر	٨٧
٢٩٨	الثامن والثمانون	٨٨
١٠٢	السادس والعشرون	٨٩
٣٠١	التاسع والثمانون	٩٠
٣٨٣	المائة والتاسع عشر	٩١
٣٧٨	المائة والسابع عشر	٩٢
٤٣٧	المائة والتاسع والثلاثون	٩٣
٤٥٥	المائة والسادس والأربعون	٩٤
٤٦٩	المائة والثاني والخمسون	٩٥
١٢٧	الثالث والثلاثون	٩٦
٣٧٣	المائة والخامس عشر	٩٧
٣٢٤	السادس والتسعون	٩٨
٢٥٠	الحادي والسبعون	٩٩
٤٣٣	المائة والسابع والثلاثون	١٠٠
١٤٧	التاسع والثلاثون	١٠١
١٢٤	الثاني والثلاثون	١٠٢
٦٥	الثالث عشر	١٠٣
١١٧	الثلاثون	١٠٤
١١١	التاسع والعشرون	١٠٥
٤٤٨	المائة والرابع والأربعون	١٠٦
١٨٩	الثاني والخمسون	١٠٧
٤٣٠	المائة والسادس والثلاثون	١٠٨
١٦١	الثالث والأربعون	١٠٩

١٢١	الحادى والثلاثون	١١٠
٣٣٣	المائة	١١١
٥٠	التاسع	١١٢
١٨٣	الخمسون	١١٣
٣٩٥	المائة والثالث والعشرون	١١٤
٤٨٢	المائة والسادس والخمسون	١١٥
٤٨٥	المائة والسابع والخمسون	١١٦
٤١٠	المائة والسابع والعشرون	١١٧
٤٦٤	المائة والخمسون	١١٨
٤٦٦	المائة والحادى والخمسون	١١٩
٨٢	العشرون	١٢٠
٩٩	الخامس والعشرون	١٢١
٣٤٩	المائة وسبعة	١٢٢
٢١٧	الستون	١٢٣
٢٤٠	الثامن والستون	١٢٤
٥٤	العاشر	١٢٥
٥٨	الحادى عشر	١٢٦
٦٠	الثانى عشر	١٢٧
٦٨	الرابع عشر	١٢٨
٨٧	الثانى والعشرون	١٢٩
٣٩٢	المائة والثانى والعشرون	١٣٠
٣٩٧	المائة والرابع والعشرون	١٣١
٢٦٤	السادس والسبعون	١٣٢
١٣٤	الخامس والثلاثون	١٣٣
٧٢	السادس عشر	١٣٤
٣٤٧	المائة وستة	١٣٥
٣٢٧	السابع والتسعون	١٣٦
٣٥١	المائة وثمانية	١٣٧

٣٦٤	المائة واثنى عشر	١٣٨
٣٢٩	الثامن والتسعون	١٣٩
٤٨٧	المائة والثامن والخمسون	١٤٠
٤٨٩	المائة والتاسع والخمسون	١٤١
٤٩٧	المائة والثاني والستون	١٤٢
٣٤٥	المائة وخمسة	١٤٣
١٩٦	الرابع والخمسون	١٤٤
٢٨٤	الرابع والثمانون	١٤٥
٢٨٢	الثالث والثمانون	١٤٦
٤٩٤	المائة والحادي والستون	١٤٧
٣٣١	التاسع والتسعون	١٤٨
٣٣٦	المائة وواحد	١٤٩
٣٣٨	المائة واثنين	١٥٠
٤٣٦	المائة والثامن والثلاثون	١٥١
٤٤١	المائة والحادي والأربعون	١٥٢
٤٤٣	المائة والثاني والأربعون	١٥٣
٢٢٠	الواحد والستون	١٥٤
٣٢٠	الخامس والتسعون	١٥٥
٢٣٨	السابع والستون	١٥٦
١٥٠	الأربعون	١٥٧
٢١٤	التاسع والخمسون	١٥٨
٨٤	الحادي والعشرون	١٥٩
١٤٢	السابع والثلاثون	١٦٠
٧٤	السابع عشر	١٦١
٢٧٦	الثمانون	١٦٢
٤١٤	المائة والتاسع والعشرون	١٦٣
٤١٧	المائة والثلاثون	١٦٤

الدرس الأول

حول هذا الكتاب

مقدمة

يصف هذا الكتاب مترجم البيسك للصفيفة الرابعة version 4.0 من البيسك السريع Quick BASIC الذى أعدته شركة ميكروسوفت Microsoft وبيئة إعداد البرامج بهذه اللغة، ويوجد لمترجم بييسك السريع سطح بينى بسيط للمستفيد وكذلك تكوين بسيط للقوائم، ويخدم تصميمه فى مساعدة المستفيد فى إعداد برامجه بأقل عدد من الخطوات، ويتأكد المنقح الذكى من التكوين مع ادخال أسطر البرنامج، ويترجم البرنامج طبقاً لما كتب عليه مع السماح بتشغيل البرنامج فى خطوة واحدة، وتعرف الأخطاء مع سردها فى تقرير بطريقة تسمح للمستفيد أن يصححها على الفور.

ويوضح بييسك السريع المشروح هنا استخدام النوال والإجراءات المختلفة المتاحة كجزء من مترجم بييسك السريع، وقد صمم الكتاب ليستخدمه المبتدئون والمهنيون كذلك، كما أنه صمم ليعلم كوسيلة للأساتذة فى قاعات التدريس، ويفترض هذا الكتاب أن هناك معرفة بالعمل باستخدام لوحة مفاتيح الكمبيوتر ونظام التشغيل DOS من قبل القارئ.

التنظيم

لخدمة القاعدة العريضة من المستفيدين المختلفين المقدم لهم هو الكتاب فقد تم تنظيمه فى دروس صغيرة سهلة القراءة، وأنت تقرأ الدرس الأول الآن.

والدرس الثانى عبارة عن عرض عام للصفيفة 4.0 من بييسك السريع، ويقدم هذا الدرس معلومات عن محتويات الأقراص واقتراحات لإعداد الكمبيوتر المتاح لك لاستخدام بييسك السريع.

الدرس الثالث، وهو عينة لجلسة مع بييسك السريع، يضع يدك على التوضيح، وتستطيع من هناك ان تشعر بتكامل بييسك السريع ومنقح الشاشة الكاملة وقائمة التشغيل Run، وتتعلم كيفية عمل ملف تنفيذى EXE، من برامجك بحيث يمكنك أن تقوم بتشغيل مثل هذه البرامج بدون استخدام بييسك السريع، ولاكتساب الخبرة بعدى بساطة امكانية استخدام بييسك السريع فقد ترغب فى القفز إلى الدرس الثالث مباشرة لرؤية ذلك بنفسك.

وتناقش الدروس المتبقية الدوال والاجراءات وتكوينات اللغة المختلفة المتاحة للمستفيد من بيسك السريع. فهي تقدم أوصافاً وتطبيقات وعمليات نمطية للبرامج للمساعدة في حل مشاكل عملية يومية. كما تقدم عديداً من الأمثلة المفيدة والمتعة في هذا الكتاب والتي يمكن أن تكون وسائل مرتفعة القيمة عندما تقوم ببناء برامج أكبر وأكثر تعقيداً. وقد تم اختبار كل الأمثلة ووجد أنها تعمل كلها بطريقة صحيحة. بالنسبة للمبتدئين فهذا يساعد في التغلب على الصدمة الأولى لمحاولة استخدام دالة جديدة أو تركيبة جديدة بدون نجاح. والدروس مرتبة ترتيباً أبجدياً (طبقاً للحروف الانجليزية) لسهولة الرجوع إليها والاتصال بأوامر محددة وأمثلة برمجة معينة.

وفي بداية الكتاب يوجد تسلسل للتعليم يوصى باتباعه. فهذا التسلسل يرتب الدروس ترتيباً منطقياً للمبتدئين وذلك ليعتصموا على استخدام بيسك السريع خطوة بخطوة أو ليستخدّمه الاساتذة في قاعات الدراسة كخطوط عريضة للمقرر.

وتقدم الملاحق من A إلى G معلومات اضافية عن بيسك السريع وتمارين برمجة لكل الدروس.

ملحق A عبارة عن قائمة بالمصطلحات المستخدمة في الكتاب ووصف موجز لها. وملحق B عبارة عن مقدمة لبيئة تطوير بيسك السريع. وينصح القارئ بقراءة هذا الملحق أكثر من مرة واحدة. ويحتوى ملحق C على جدول وقائمة بكلمات بيسك السريع الرئيسية. ويناقش ملحق D مترجم أسطر الأوامر والواصل وخيارات الترجمة المختلفة المتاحة. ويناقش ملحق E إنتاج وصيانة مكتبات بيسك السريع. ويسرد ملحق F رسائل الخطأ التي تنتج بواسطة بيسك السريع. كما يقدم ملحق G تمارين مصممة لاختبار المعلومات التي تم تعلمها من هذه الدروس.

متطلبات نظم المكونات ونظم البرامج

يحتاج مترجم بيسك السريع من شركة ميكروسوفت إلى ما يلي :

- جهاز كمبيوتر IBM PC أو أى جهاز آخر متوافق معه تماماً ويستخدم نظام تشغيل MS-DOS أو PC-DOS الصيغ من 2.0 فأعلى منها.
 - مشغل أقراص مرنة واحد على الأقل ويفضل وجود مشغلين.
 - ذاكرة اتصال عشوائى (RAM) لا تقل عن 320 كيلوبايت.
- انتقل إلى الدرس الثانى للاستمرار فى تسلسل التعلم الموصى به.

الدرس الثانى

عرض عام لبيسك السريع

مقدمة

لقد كانت شركة ميكروسوفت رائدة فى تطوير نظم البرامج المبتكرة لأجهزة الميكروكمبيوتر. وفى تقليدها هذا فإن ببسك السريع (الصيغة 4.0) هو منتج يمثل نقطة تحول ويقدم بيئة إعداد برامج متكاملة ومذهلة بلغة البيسك.

وتشمل البيئة شاشة كاملة ومنقحاً للتأكد من التكوين وامكانيات التنقيح للملفات عديدة ونوافذ عديدة وتسهيلات للتصحيح الكامل وقوائم للسحب لأسفل وتكوين قائمة سهل وقوى يمكن قيادته من خلال لوحة المفاتيح أو باستخدام فأرة. وبمجرد أن تتواجد فى بيئة ببسك السريع فيمكنك أن تنقح البرنامج وتقوم بتشغيله وتصحيحه بدون أن تغير من البرنامج أو الأقراص.

وبيسك المفسر يكون متسعاً فى هذا التفسير للبيسك فى عديد من الأوجه :

- تسمح لك البرامج الفرعية بتجزئة البرامج إلى أجزاء فردية ومستقلة منطقياً.

- دعم البرمجة المرتبة مقدم فى صورة IF.. THEN متعددة الأسطر ومكونات تحكم مساراً مثل WHILE.. WEND و SUB و إجراءات FUNCTION وعبارات TYPE.. END TYPE لتغييرات يعرفها المستفيد وإجراءات إعادة ذاتية.

- يمكن استخدام عناوين حرفية عديدة تجعل البرنامج أكثر سهولة فى قراءته وكتابته لأن ببسك السريع لا يتطلب أرقاماً للأسطر الخاصة بعبارات برامجه.

- يمكن اقرار المنظومات الديناميكية لاستغلال الذاكرة استغلالاً أمثل.

- يسمح بمنظومات عديدة كبيرة، فيمكن استخدام أى عدد من منظومات حجمها 64K والتي تتسق مع الذاكرة المتاحة.

يسمح بأن تقوم ما تسمى بأشباه الأوامر metacommands بالتحكم فى الطريقة التى يفسر ويترجم بها المترجم برنامجك. مثال ذلك، يجعلك الأمر \$INCLUDE تدخل ملفات مصدر إضافية أثناء الترجمة.

محتويات الأقراص :

يأتى بيسك السريع (الصفحة 4.0) فى ثلاثة أقراص مسعاة بقرص البرنامج وقرص المكتبات وقرص المنافع والأمثلة على التوالى. وفيما يلى قائمة بمحتويات كل قرص :

القرص الأول : قرص البرنامج

Disk 1

Filename	Description
BC.EXE	(BASIC command-line compiler)
QB.EXE	(QuickBASIC program)
SETUP.BAT	(QuickBASIC installation program)
SETUP1.BAT	(" " " ")
PACKING.LST	(List of files on the disks)
README.DOC	(Latest information on QuickBASIC)

القرص الثانى : قرص المكتبات

Disk 2

Filename	Description
BRUN40.EXE	(QuickBASIC run-time module)
BRUN40.LIB	(QuickBASIC-library run-time support)
BCOM40.LIB	(QuickBASIC alternate run-time library)
BQLB40.LIB	(Quick-library run-time support)

القرص الثالث : قرص المنافع والأمثلة

Disk 3

Filename	Description
SOURCE	< DIR >
LIB.EXE	(Microsoft library manager)
LINK.EXE	(Microsoft overlay linker)
QB.HLP	(Help text for QuickBASIC)
MOUSE.COM	(Mouse driver)
NOEM.OBJ	(Programming support file)
REMLINE.BAS	(Example program)
SORTDEMO.BAS	(" ")
TORUS.BAS	(" ")
ABSOLUTE.ASM	(" ")
INTRPT.ASM	(" ")
QB.LIB	(QuickBASIC library; supports DOS interrupts)
QB.QLB	(Quick-library; supports DOS interrupts)
QB.PIF	(QuickBASIC system file)
QB.BI	(" " " ")
NOCOM.OBJ	(Programming support file)
DEMO1.BAS	(" ")
DEMO2.BAS	(" ")
DEMO3.BAS	(" ")
QBHERC.COM	
FIXSHIFT.COM	(" ")

Contents of Directory Source

Filename	Description
INDEX.BAS	(Example program)
CAL.BAS	(")
CUBE.BAS	(")
EDPAT.BAS	(")
MANDEL.BAS	(")
SINEWAVE.BAS	(")
BAR.BAS	(")
PALETTE.BAS	(")
BALJPSET.BAS	(")
WHEREIS.BAS	(")
CRLF.BAS	(")
BALLYOR.BAS	(")
COLORS.BAS	(")
FILERR.BAS	(")
PLOTTER.BAS	(")
STRTONUM.BAS	(")
TERMINAL.BAS	(")
CHECK.BAS	(")
QLBDUMP.BAS	(")
SEARCH.BAS	(")
FLPT.BAS	(")
TOKEN.BAS	(")
ENTAB.BAS	(")
HIDE.BAS	(")

وقد تحتوى أقراصك على أمثلة أخرى وبرامج أخرى للأغراض العامة ويكون لكل برامج البيسك التوسع BAS. اقرأ الملف PACKING.LIST لمعرفة المزيد عن محتويات الأقراص.

الإعدادات الموصى به

اعتماداً على تشكيل نظام الكمبيوتر المتاح لك يمكن إعداد بيسك السريع بطرق عديدة. وفيما يلي طرق إعداد يوصى بها للثلاثة تشكيلات الممكنة :

نظام به مشغلين أقراص مرنة : لاستخدام بيسك السريع مع نظام قرصين مرنيين اتبع الخطوات التالية :

* استخدم المشغل B فى تخزين ملفات المصدر بالبيسك.

* استخدم المشغل A لبرامج بيسك السريع.

* قم بتشغيل بيسك السريع من المشغل B.

يضمن لك هذا أن ببسك السريع يستطيع أن يجد أى مكتبات قد يحتاج إليها برنامجك وتكتب ملفات المصدر والتشغيل فى المشغل B.

– نظام مشغل أقراص مرنة واحد : هناك طريقتان :

الطريقة الأولى : خزن الملفات اللازمة لتشغيل ببسك السريع (مثل QB.EXE و QB.HLP و COMMAND.COM) وملفات المصدر بالببسك على قرص واحد.

الطريقة الثانية : لاستخدام ببسك السريع قم بتشغيل ببسك السريع من قرص البرنامج. وبعد تحميل ببسك السريع استبدل قرص البرنامج بقرص فارغ واستخدم هذا القرص فى تخزين برامجك المكتوبة بالببسك.

– نظام قرص صلب : حيث إن تنظيم القرص الصلب يعتمد داخلياً على الاستخدام الفردى له فما يلى هو طريقة واحدة فقط لاعداد الادلة لاستخدام ببسك السريع :

* قم بإعداد دليل directory (QB) من دليل الجذر root directory.

* استخدم هذا الدليل لكل ملفات برامج ببسك السريع وبرامجك.

اجعل المسار DOS PATH يحتوى على دليل ببسك السريع QB\ C: فى مسار بحثه. مثال ذلك :

Set Path=C:\;C:\QB;

والبديل لذلك هو أنه يمكنك استخدام ملف دفعة للاعداد (SETUP1.BAT, SETUP.BAT) لوضع ببسك السريع على القرص الصلب.

انتقل إلى الملحق B للاستمرار فى تسلسل التعلم.

الدرس الثالث

عينة لجلسة مع بيسك السريع

مقدمة

تقدمك عينة الجلسة هذه لمترجم بيسك السريع من شركة ميكروسوفت وتعودك هذه الجلسة على السطح البيئي للمستفيد وتكوين قائمة البيئة المتكاملة للبيسك السريع.

وفى هذه الجلسة تقوم بأداء ما يلى :

- كتابة برنامج فى منتج بيسك السريع.
- تنفيذ البرنامج الذى كتبته.
- تتبع الاختيارات الأخرى للشاشة.
- حفظ البرنامج الذى كتبته.
- انتاج ملف تنفيذ EXE.
- الخروج من بيسك السريع.

إذا ما كان لهذا وقع بأنه عمل كبير فلا تهتم بذلك. فلن تستغرق الجلسة أكثر من 20 دقيقة فقط.

البداية

يفترض أنك معتاد بدرجة كافية على تشغيل جهاز الكمبيوتر المتاح لك.

نظام قرصين مرنيين : إذا كان لديك نظام قرصين فاستمر على النحو التالى :

- ١ - ادخل قرص المترجم رقم 1 فى المشغل A.
- ٢ - ادخل قرص العمل (قرص مشكل فارغ) فى المشغل B.
- ٣ - من المشغل B اكتب A:QB واضغط على مفتاح الإدخال.

نظام القرص الصلب : إذا كان لديك نظام قرص صلب (قرص ثابت) فاستمر على النحو

التالى:

ملاحظة

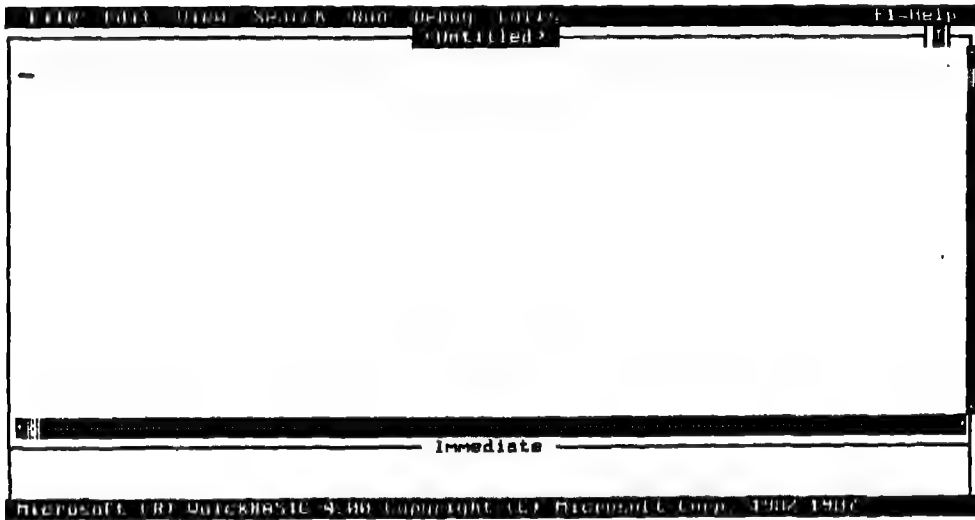
إذا لم تكن قد انتجت دليل بيسك السريع بالفعل ونسخت كل ملفات البرامج فيه فافعل ذلك

الآن.

١ - غير الدليل المفتوح إلى دليل بيسك السريع وذلك بكتابة CD\QB والضغط على مفتاح الإدخال.

٢ - اكتب QB واضغط على مفتاح الإدخال.

لاحظ أن شاشة بيسك السريع تظهر في عدة ثوان. ويكون منقح بيسك السريع متاحاً لك لإدخال البرنامج وتشغيله. لاحظ كذلك خيارات القائمة الموجودة في قعة الشاشة والنافذة الفورية الموجودة في قاعدة الشاشة ونقطة البداية الموجودة في الركن العلوي الأيسر.



إدخال البرنامج : منقح بيسك السريع عبارة عن شاشة كاملة وحساس للمحتوى والتكوين ووسيلة تحكم ذاتية لكتابة البرامج. واستخدام المنقح هو أحد الأنشطة البسيطة. لإدخال البرنامج ابدأ في كتابته ببساطة. واضغط على مفتاح الإدخال عند الانتهاء من كل سطر. فإذا لم يسبق لك استخدام منقح شاشة كاملاً (مثل Wordstar) فقد تكون في حاجة، على ذلك، للرجوع إلى الملحق B للتعرف على منقح بيسك السريع. اكتب الآن البرنامج التالي :

```

File Edit View Search Run Debug Calls F1=Help
BOX.BAS

'Constants
CONST ulx = 10, uly = 5, Lrx = 40, Lry = 10
CONST horiz = 195, Vert = 179, trc = 191, tlc = 218, brc = 217, blc = 192

'Variables
Lin% = 1
WStr$ = ""

GOTO Start

DrawBox:
LOCATE uly, ulx: PRINT CHR$(tlc)
LOCATE Lry, ulx: PRINT CHR$(blc)
LOCATE uly, Lrx: PRINT CHR$(trc)
LOCATE Lry, Lrx: PRINT CHR$(brc)
LOCATE uly, ulx + 1: PRINT STRING$(Lrx - ulx - 1, horiz):
LOCATE Lry, ulx + 1: PRINT STRING$(Lrx - ulx - 1, horiz):
FOR Cnt = uly + 1 TO Lry - 1
    LOCATE Cnt, ulx: PRINT CHR$(Vert)
    LOCATE Cnt, Lrx: PRINT CHR$(Vert)
NEXT Cnt
RETURN

WriteInBox:
IF LEN(WStr$) > Lrx - ulx - 2 THEN GOTO WriteRet
IF Lin% > Lry - uly - 2 THEN GOTO WriteRet

Tb% = ((Lrx - ulx - 2) - LEN(WStr$)) / 2
LOCATE uly + Lin%, ulx + Tb%: PRINT WStr$:

WriteRet:
RETURN

Start:
CLS
GOSUB DrawBox
WStr$ = "WELCOME TO"
GOSUB WriteInBox
Lin% = Lin% + 1
WStr$ = "Illustrated QuickBASIC"
GOSUB WriteInBox

END

Immediate
Data: (Untitled) Context: Program not running 0000:0000

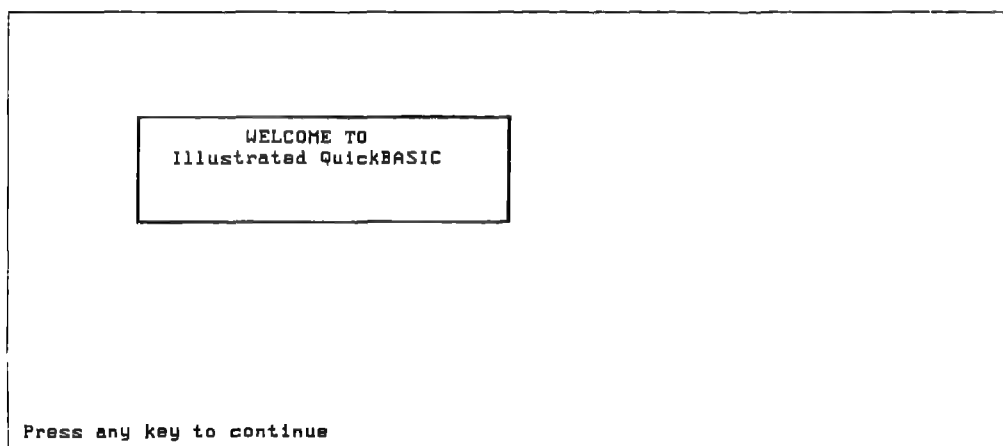
```

تشغيل البرنامج : بعد انتهائك من كتابة البرنامج استمر على النحو التالي :

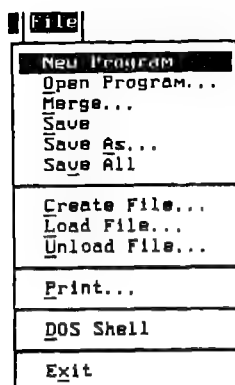
ملاحظة :

بالنسبة إلى تسلسل المفاتيح المتصل بشرطة (مثل Ctrl-C) اضغط على المفتاح الأول واستمر في الضغط عليه أثناء الضغط على المفتاح الثاني.

- ١ - اضغط على Shift-F5 لتشغيل البرنامج. فإذا ما اكتشف بيسك السريع أى أخطاء فلن يعمل البرنامج. تحت هذه الظروف يزيد بيسك السريع من اضاءة سطر البرنامج الموجود به خطأ ويظهر نافذة على الشاشة تصف الخطأ. فإذا ما حدث ذلك لك فاستمر على النحو التالى :
 أ - لاحظ الخطأ الموجود على الشاشة. اضغط على قضيب المسافات Spacebar للتخلص من نافذة رسالة الخطأ.
- ب - افحص قائمة البرنامج الموجودة فى الدرس واكتب العبارة الصحيحة.
- ج - اضغط على Shift-F5 لتشغيل البرنامج.
- ٢ - لاحظ مخرجات البرنامج على الشاشة.



- ٣ - اضغط على قضيب المسافات للعودة إلى منقح بيسك السريع.
- فحص القوائم : افحص بسرعة قوائم بيسك السريع الأخرى على النحو التالى :
 ١ - اضغط على Alt-F لعرض قائمة الملفات.
- ٢ - اضغط على مفتاح السهم الأيمن لعرض القوائم الأخرى.
- ٣ - اضغط على Esc للعودة إلى المنقح.
- حفظ البرنامج : احفظ برنامجك الآن بحيث يمكنك استخدامه مع سمات بيسك السريع الأخرى فيما بعد.
- ١ - اضغط على Alt-F للحصول على قائمة الملفات.



لاحظ الثلاث خيارات للحفظ. يحفظ خيار Save الملف بدون أى مزيد من اللفظ، فإذا كان الملف جديداً فيسالك الخيار عن اسم الملف. خيار Save As يسمح لك بحفظ الملف تحت اسم جديد. سوف تستخدم هذا الخيار بكثرة فى دروس لاحقة، ويحفظ خيار Save All كل الملفات الموجودة حالياً فى الذاكرة.

٢ - اكتب A لاختيار Save As.

٣ - اكتب BOX.BAS كاسم للملف. لحفظ البرنامج على مشغل مختلف أضف مؤشر المشغل إلى اسم الملف. مثال ذلك B:BOX.BAS.

٤ - اضغط على Tab ومفتاح السهم السفلى لاختيار ملف نص ASCII.

٥ - اضغط على Tab مرة أخرى ثم على قضيب المسافات لحفظ الملف.

انتاج ملف تنفيذ EXE: يمكن تنفيذ الملف الذى كتبته بدون استخدام بيسك السريع. لعمل ذلك يجب عليك أن تترجم هذا البرنامج وتحفظه على القرص. قائمة تنفيذ Run بيسك السريع يوجد بها اختيار يترجم البرنامج ويحفظه على القرص. والصيغة المترجمة يكون لها الاتساع EXE. ويمكن تنفيذ هذه الصيغة من DOS ببساطة عن طريق كتابة اسم الملف مثلما يلى :

A:>BOX

لتجربة ذلك استمر على النحو التالى :

١ - اضغط على Alt-R واختر الخيار Make EXE file.

٢ - لاحظ المستطيل الذى يظهر على الشاشة مع خيارات أكثر ووجود اسم الملف BOX.EXE الذى يكون هناك بالفعل. (اضغط على Tab ثلاث مرات ثم اضغط على قضيب المسافات. وينقلك هذا إلى الخيار Stand Alone EXE file.

٣ - اضغط على Tab مرة أخرى لاختيار الخيار Make EXE عند قاعدة المستطيل واضغط على قضيب المسافات.

يترجم بيسك السريع البرنامج ثم يصله ويكتبه على القرص. فإذا كانت لديك أى مشاكل أثناء هذه العملية فتأكد أن بيسك السريع قد سبق إعداده بطريقة صحيحة. ولزيد من المعلومات عن إعداد بيسك السريع أرجع إلى الدرس الثانى.

لتشغيل ملف EXE من DOS استمر على النحو التالى :

١ - اضغط على Alt-F ثم اكتب D. يسمح لك هذا الخيار بتنفيذ أوامر DOS بدون وجود بيسك السريع.

٢ - اكتب BOX عند ملقن DOS واضغط على مفتاح الادخال. لاحظ أن البرنامج ينفذ.

٣ - اكتب Exit للعودة إلى بيسك السريع.

الخروج من بيسك السريع : عندما تنتهى من عملك كله وتريد الخروج من بيسك السريع لأداء أى عمل آخر استمر على النحو التالى :

١ - اضغط على Alt-F ثم اكتب X للخروج من منقح بيسك السريع.

٢ - انتقل إلى الدرس الرابع للاستمرار فى تسلسل التعلم.

الدرس الرابع

الترجمة من DOS

هناك طريقة أخرى لترجمة البرنامج الذى أعدته فى الدرس الثالث وتنفيذه وهى ترجمته من ملقن DOS باستخدام مترجم سطر الأمر BC.EXE، وتتكون هذه العملية من خطوتين :

١ - ترجمة

٢ - وتوصيل

أثناء خطوة الترجمة تترجم عبارات برنامج البيسك إلى لغة الآلة وتكتب فى ملف مع اسم البرنامج والتوسع OBJ. وفى خطوة التوصيل يضاف مقطع المكتبة المناسب إلى شفرة الآلة ويتم انتاج برنامج قائم بذاته وكتابته فى ملف له نفس اسم الملف ولكن بالتوسع EXE، ويمكن أن ينفذ هذا الملف بكتابة اسمه عند ملقن DOS والضغط على مفتاح الادخال.

للمزيد من المعلومات عن ترجمة سطر الأوامر وعملية الاتصال ارجع إلى الملحق D.

عندما تقوم بترجمة البرنامج BOX.BAS من بييسك السريع يجب أن تكون قد لاحظت أن بييسك السريع قد انتج العديد من أوامر DOS ونفذها، وفى هذا الدرس تقوم بكتابتها بنفسك وهى تنفذ.

وأسباب عمل ذلك بهذه الطريقة هى ما يلى :

- قد يكون البرنامج كبيراً جداً لترجمته من داخل بييسك السريع.
 - قد تكون هناك حاجة إلى خيارات سطر الأوامر.
 - يمكن أن يكتب البرنامج فى صورة مضغوطة بمصحح Code View.
 - لانتاج برنامج ينفذ ومتوافق مع مصحح Code View.
 - لانتاج قائمة ملف من الترجمة.
 - لتوصيل OBJ. NOCOM الذى يقلل حجم ملفات EXE. التى لاستخدم عبارة COM.
- وقد تم تقديم ذلك هنا لإعطاء القارئ الشعور بالامكانيات المصاحبة لبييسك السريع. وسوف تقوم فى هذا الدرس بتنفيذ الأنشطة التالية بصورة مبسطة :

- ترجمة BOX.BAS من DOS باستخدام BC.EXE.
- توصيل BOX.OBJ ونتاج BOX.EXE باستخدام LINK.EXE.
- تنفيذ BOX.EXE.

الترجمة

تقوم في هذه العملية بترجمة برنامج BOX.BAS مستخدماً مترجم بيسك السريع القائم بذاته، ابدأ عند ملقن DOS.

نظام قرص مرّن واحد : إذا كنت تستخدم جهاز كمبيوتر به مشغل أقراص واحد فقط فاستمر على النحو التالي :

- ١ - ادخل قرص بيسك السريع واكتب BC ثم اضغط على مفتاح الإدخال.
 - ٢ - استبدل قرص البرنامج بالقرص الموجود عليه برامجك، اكتب BOX.BAS كاسم لملف المصدر واضغط على مفتاح الإدخال، اضغط على مفتاح الإدخال كإجابة على بقية الملقنات.
- نظام قرصين مرّنين : إذا كان لديك مشغلاً أقراص فاستمر على النحو التالي :

- ١ - ادخل قرص بيسك السريع في المشغل A وقرص برامجك في المشغل B واجعل المشغل B هو المشغل المستخدم.
- ٢ - اكتب A:BC BOX.BAS/d/o; واضغط على مفتاح الإدخال.

- نظام قرص صلب : إذا كان لديك قرص صلب فاستمر على النحو التالي :
- ١ - تأكد من وجودك في دليل بيسك السريع بكتابتك CD\QB والضغط على مفتاح الإدخال.
 - ٢ - اكتب BC BOX.BAS/d/o; واضغط على مفتاح الإدخال.

التوصيل

في هذا القسم تقوم بتوصيل ملف OBJ، مع مكتبات لدعم وقت تنفيذ بيسك السريع، ويحدث ذلك طبقاً للطريقة التالية :

- نظام قرص مرّن واحد : يحدث التوصيل مع قرص مرّن واحد فقط على النحو التالي :
- ١ - انسخ الملفات التالية على قرص مشكل وفارغ :

- BCOM40.LIB
- BOX.OBJ

- ٢ - ادخل قرص بيسك السريع فى المشغل واكتب LINK واضغط على مفتاح الادخال.
- ٣ - استبدل قرص بيسك السريع بالقرص الذى انتجته فى الخطوة الاولى. اكتب BOX واضغط على مفتاح الادخال ثم اضغط على مفتاح الادخال مرة أخرى كرد على بقية الملقنات.
- نظام بمشغلى أقراص مرنة : يتم التوصيل بمشغلين على النحو التالى :
- ١ - إعداد قرص كما هو مذكور فى الخطوة الأولى من قسم المشغل الواحد.
- ٢ - ادخل قرص بيسك السريع فى المشغل A والقرص الناتج فى الخطوة الأولى فى المشغل B. اجعل المشغل B هو المشغل المستخدم وذلك بكتابة : B والضغط على مفتاح الادخال.
- ٣ - اكتب BOX.EXE, LINK BOX.OBJ, ثم اضغط على مفتاح الادخال.

تنفيذ البرنامج

لتنفيذ البرنامج الذى سبق لك ترجمته اتبع ما يلى :

نظام قرصين مرنيين :

- ١ - ادخل القرص الموجود فيه "BOX, EXE" فى المشغل A.
- ٢ - غير المشغلات إلى A واكتب BOX واضغط على مفتاح الادخال.

نظام قرص صلب :

- ١ - إذا لم تكن فى الدليل "C:\QB" فانتقل إليه.
 - ٢ - اكتب BOX واضغط على مفتاح الإدخال.
- انتقل إلى الدرس الثالث والعشرين للاستمرار فى تسلسل التعلم.

الدرس الخامس

دالة القيمة المطلقة ABS

الوصف

تعيد دالة القيمة المطلقة ABS قيمة التعبير العددي المطلقة وتكوينها هو كما يلي :
(تعبير عددي) ABS

وتعود قيمة التعبير العددي كقيمة صحيحة بدون اشارة.

التطبيقات

تستخدم دالة ABS في الحصول على الجزء الصحيح لتعبير عددي بدون اشارة. في بعض الاحيان لا تكون اشارة العدد مهمة في اتخاذ القرارات وتصبح دالة ABS مفيدة في هذه الاحوال، وفيما يلي بعض الامثلة.

مثال ١

```
X = 38.3 : Y = -99
Z = ABS(X-Y)
```

مثال ٢

```
QuadRoot# = 1.316074      ' square root of square root of 3
PRINT ABS(QuadRoot#)
```

عملية تقليدية

توضح دالة ABS في العملية التقليدية التالية. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Exit F1=Help
<Untitled>
'Program to iteratively take the square root until the result approaches zero
PRINT "Enter a number to get the square root."
PRINT "A negative number ends the program."

INPUT "Enter number "; x
IF x <= 0 THEN END
Iter% = 1: PSqr = SQR(x)
DO WHILE ABS(1 - PSqr) >= .00001
    PSqr = SQR(PSqr)
    Iter% = Iter% + 1
LOOP
PRINT "The square root of the number was taken until it approached zero."
PRINT "Number of iterations: "; Iter%

----- Immediate -----
Main: <Untitled> Context: Program not running 0001:0:00

```

٢ - نفذ البرنامج ولاحظ استخدام دالة ABS في البرنامج.

```

Enter a number to get the square root.
A negative number ends the program.
Enter number ? 233
The square root of the number was taken until it approached zero.
Number of iterations: 28

Press any key to continue

```

٣ - اضغط على أى مفتاح للعودة إلى البرنامج. اختر Save من قائمة الملفات واكتب ABS.

BAS كاسم للملف وحدد أن صيغة الملف نصية واحفظ البرنامج.

٤ - من قائمة الملفات اختر New مع اخلاء الشاشة.

٥ - انتقل إلى الدرس المائة والأربعين للاستمرار فى تسلسل التعلم.

الدرس السادس

دالة ASC

الوصف

تعيد دالة ASC قيمة ASCII لأول رمز من تعبير السلسلة، وتكوينها هو ما يلي :
(تعبير سلسلة) ASC
إذا كان تعبير السلسلة فارغاً فينتج المترجم رسالة خطأ وقت التشغيل.

التطبيقات

دالة ASC مفيدة جداً في الحصول على قيمة ترتيبية من جدول ASCII لرمز معين. ويمكن أن تستخدم ضمن أشياء أخرى في تحويل تعبيرات السلاسل للأعداد إلى قيم عددية ثم تحويل السلاسل بعد ذلك إلى الحروف العلوية (الكبيرة) أو السفلية (الصغيرة).

```
ASC("A")
```

وتعيد الدالة :

قيمة ASCII للحرف "A"

```
ASC("Hello")
```

كما تعيد الدالة :

قيمة ASCII للحرف "H"

والبرنامج :

```
Exp$ = "Prep"
FOR Cnt = 1 TO LEN(Exp$)
    PRINT ASC(MID$(Exp$,Cnt,1)) " ";
NEXT Cnt
```

يطبع قيمة ASCII لكل رمز موجود في السلسلة Exp\$.

عملية تقليدية

يحول المثال التالي أحد تعبيرات السلسلة إلى قيمة عددية. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :


```

File Edit View Search Run Debug Calls F1=Help
Untitled:
Num$ = "1234": Num% = 0
FOR i = 1 TO LEN(Num$)
    Num% = Num% + ((ASC(MID$(Num$, i, 1)) - 48) * (10 ^ (LEN(Num$) - i)))
NEXT i
PRINT Num$, Num%

Immediate
Data: Untitled: Controls: Program not running 00000:017

```

٢ - نفذ البرنامج ولاحظ تحويل السلسلة Num\$ إلى مكافئها العددي. لاحظ كذلك استخدام قيمة ASCII الناتجة من دالة ASC

```

1234      1234

Press any key to continue

```

٣ - اضغط على أى مفتاح للعودة إلى البرنامج.

٤ - اختر Save من قائمة الملفات واكتب ASC.BAS كاسم للملف. حدد أن تشكيل الملف نصي واحفظ الملف.

٥ - انتقل إلى الدرس الثانى والستين للاستمرار فى تسلسل التعلم.

الدرس السابع

دالة ATN

الوصف

تعيد دالة ATN قوس الظل لتعبير عددي. وقوس الظل هو الزاوية التي يكون ظلها مساوياً للتعبير العددي. وتكوينها هو كما يلي :

ATN (تعبير عددي)

تكون القيمة الناتجة بالتقدير الدائري وتقع في العددي من $-\pi/2$ إلى $+\pi/2$ (حيث $\pi = 3.141593$). وتقوم هذه القيمة كعدد له دقة فردية كاصطلاح تقليدي أما إذا كان التعبير العددي له دقة مزدوجة فتعود قيمة قوس الظل بدقة مزدوجة. ويكون التعبير العددي من أي نوع من الأنواع العددية.

التطبيقات

تستخدم دالة ATN عندما تكون هناك حاجة إلى قوس الظل لأحد الأعداد كما في حالة تشغيل الأعداد أو تطبيقات الألعاب. وفيما يلي بعض الأمثلة :

```
P1 = 3.141592653
At = ATN(P1/1.5)
PRINT ATN(P1/3)
```

عملية تقليدية

توضح العملية التالية دالة ATN. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls
Untitled
This program demonstrates the ATN function.
CLS
PRINT "The program computes polar coordinates from rectangular"
PRINT "coordinates."
PRINT
INPUT "Enter x and y separated by commas "; x, y
IF x + y = 0 THEN GOTO 111
IF x = 0 THEN GOTO 222
IF y = 0 THEN GOTO 333

AngleDeg = ATN(y / x)
AngleDeg = (AngleDeg * 180) / 3.14159
Mag = SQR(x ^ 2 + y ^ 2)
GOTO PrintLine

111 :
    AngleDeg = 0: Mag = 0
    GOTO PrintLine

222 :
    IF y > 0 THEN
        AngleDeg = ABS(y)
    ELSE
        AngleDeg = -90
        Mag = ABS(y)
    END IF
    GOTO PrintLine

333 :
    IF x > 0 THEN
        AngleDeg = 0: Mag = x
    ELSE
        AngleDeg = 180
        Mag = ABS(x)
    END IF

PrintLine:
PRINT "Polar coordinates: Angle in degrees = "; AngleDeg
PRINT "Magnitude = "; Mag

Immediate
Main: Untitled  Code: Program not running  00017-023

```

٢ - نفذ البرنامج. لاحظ النتائج واستخدام دالة ATN في البرنامج.

The program computes polar coordinates from rectangular coordinates.

Enter x and y separated by commas : 10,10
Polar coordinates: Angle in degrees = 45.00004
Magnitude = 14.14214

Press any key to continue

- ٣ - اضغط على أى مفتاح وعد إلى البرنامج، اختر Save من قائمة الملفات واكتب ATN.BAS
كاسم للملف وحدد أن صيغة الملف نصية واحفظ البرنامج.
- ٤ - انتقل إلى الدرس الرابع والعشرين للاستمرار فى تسلسل التعلم.

الدرس الثامن

عبارة BEEP

الوصف

تنتج عبارة BEEP صوتاً من مكبر الصوت الموجود في الكمبيوتر. وتكوينها هو كما يلي :

BEEP

التطبيقات

عبارة BEEP مفيدة في شد انتباه المستفيد إلى النشاط الحالي في البرنامج. وفيما يلي

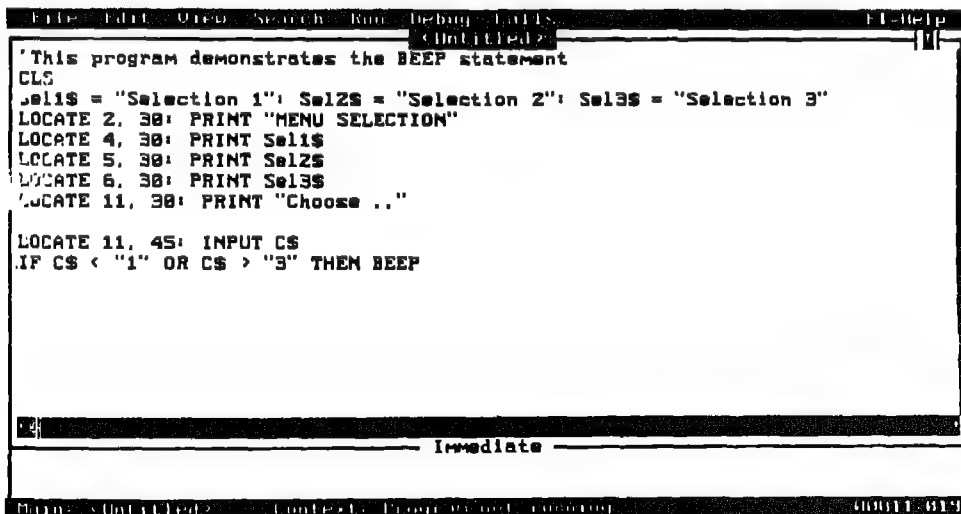
بعض الأمثلة :

```
IF (INKEY$ = "") THEN BEEP
IF (Cnt% > 3) THEN BEEP
IF ErrSignal THEN BEEP
BEEP:BEEP:BEEP: PRINT "No more records !"
```

عملية تقليدية

توضح هذه العملية استخدام عبارة BEEP. ابدأ بتحميل ببسك السريع.

١ - أكتب البرنامج التالي :



```
File Edit View Search Run Debug Exit
Untitled2
This program demonstrates the BEEP statement
CLS
Sel1$ = "Selection 1": Sel2$ = "Selection 2": Sel3$ = "Selection 3"
LOCATE 2, 30: PRINT "MENU SELECTION"
LOCATE 4, 30: PRINT Sel1$
LOCATE 5, 30: PRINT Sel2$
LOCATE 6, 30: PRINT Sel3$
LOCATE 11, 30: PRINT "Choose .."

LOCATE 11, 45: INPUT CS
IF CS < "1" OR CS > "3" THEN BEEP

Immediate
Run: (Untitled2) Context: Program not running 00011-019
```

٢ - نفذ البرنامج. اكتب 4 واضغط على مفتاح الإدخال. لاحظ استخدام عبارة BEEP في البرنامج. اضغط على أى مفتاح للعودة إلى البرنامج.

```

MENU SELECTION

Selection 1
Selection 2
Selection 3

Choose ..      ? 4

Press any key to continue
    
```

٣ - اختر New من قائمة الملفات واكتب N لإخلاء الشاشة.

٤ - انتقل إلى الدرس المائة والثانى والثلاثين للاستمرار فى تسلسل التعلم.

الدرس التاسع

عبارتا BLOAD و BSAVE

الوصف

عبارة BLOAD : تقوم العبارة بتحميل صورة من الذاكرة للـ file spec أو لوحدة من فرع محدد من الذاكرة. وتكوينها هو كما يلي :

BLOAD file spec, offset

جزء file spec هو نفسه مثل مواصفات الملف المقدمة للأوامر الأخرى لتشغيل الملفات مثل FILES و KILL و NAME AS. وتوضع أسماء المسارات وأسماء الملفات بين علامات تنصيص وتتبع في تكوينها نفس التكوين الذي تتبعه أسماء المسارات وأسماء الملفات في نظام تشغيل DOS. وجزء offset هو فرع القطاع الذي سوف تحمل البيانات فيه. والقطاع هو آخر DEF SEG ينفذ أو قطاع ببسك السريع كحالة تقليدية.

عبارة BSAVE : تنسخ العبارة البيانات من الذاكرة كما هي تماماً في ملف محدد أو مشغل معين. وتكوينها هو كما يلي :

BSAVE file spec, offset, length

جزء file spec و offset هما مثل نظيريهما في العبارة السابقة ويحدد جزء length عدد البايت الذي ينسخ من الذاكرة. وهو رقم يقع بين 0 و 65,535. وعلى هذا فالملف الذي ينتج هو نسخة من الذاكرة بايت بعد بايت مع معلومات تحكم ببسك السريع.

كل وحدات المدخلات باستثناء لوحة المفاتيح تدعم بواسطة هاتين العبارتين. ولا تدعم العبارتان من وحدة تشغيل الشرائط.

التطبيقات

هاتان العبارتان هما من العبارات المطورة المستخدمة في التعامل مع ذاكرة وقت تنفيذ ببسك السريع. ويجب استخدامهما بكل حذر. ولا تختبر عبارة BLOAD الذاكرة التي تنقل البيانات إليها وبالتالي فمن الممكن أن تكتب على البيانات الموجودة قبل النقل. وإذا ما حدث ذلك فتكون النتائج غير متوقعة. عند تحميل شاشات من ملف إلى داخل الذاكرة فمن الضروري أن تكون

حالة الشاشة هي نفسها مثلما كانت عليه عندما حدث حفظ للشاشة. أى أنه إذا كانت الشاشة قد انتجت وحفظت وهي في حالة الرسومات فيجب أن تكون في نفس حالة الرسومات كذلك عندما يتم تحميلها مرة أخرى. ويجب ألا تحمل ملفات سبق انتاجها باستخدام بيسك المطور BASICA وذلك بعبارة BLOAD لأن بيسك السريع يحمل البرامج ويخزن البيانات في مواقع مختلفة مما يفعله بيسك المطور. وفيما يلي أمثلة للعبارتين :

مثال ١

```
DIM W(10,10)
DEF SEG = VARSEG(W(1))
BSAVE "W.Dat", VARPTR(W(1)), 404
```

مثال ٢

```
DEF SEG = &hb000
BLOAD "Screen.001"
```

المثال السابق يحمل صورة للشاشة داخل ذاكرة الشاشة. ويمكن أن تستخدم هذه الطريقة في تحميل شاشات سبق انتاجها باستخدام برامج أخرى، ويقلل هذا من الشفرة اللازمة لانتاج الشاشة أثناء وقت التنفيذ.

عملية تقليدية

تستخدم هذه العملية العبارتين السابقتين في حفظ وتحميل منظومة عديدة. يتم انتاج المنظومة ووضع البيانات فيها كقيم ابتدائية ثم تخزين المنظومة في ملف ثم تمحي وبعاد عمل الأبعاد لها ثم تحمل من الملف. ويطبوع عدد محدود من عناصر المنظومة لتوضيح أنها تعمل.

ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls
BLSAVE.BAS
F1-Help

'This is a demonstration of the BLOAD and BSAVE statements.

'SDYNAMIC
DIM w(10, 10)
DEF SEG = VARSEG(w(1, 1))
FOR i = 1 TO 10
    FOR j = 1 TO 10
        w(i, j) = i + j
    NEXT j, i

'Note the length provided is the number of elements in array w plus 1
'multiplied by 4.

BSAVE "U.Dat", VARPTR(w(1, 1)), 404
FOR i = 1 TO 10
    FOR j = 1 TO 10
        w(i, j) = 0
    NEXT j, i
ERASE w
REDIM w(10, 10)

BLOAD "U.Dat", VARPTR(w(1, 1))
CLS

FOR i = 1 TO 2
    FOR j = 1 TO 10
        PRINT (w(i, j))
    NEXT j, i

END

----- Immediate -----

Title: BLSAVE.BAS      Location: Program not running      00000001

```

٢ - نفذ البرنامج. لاحظ استخدام عبارتي هذا الدرس. وتكون المخرجات على النحو التالي :

13
13
13
13
13
13
13
13
13
13
14
14
14
14
14
14
14
14
14

Press any key to continue

٣ - ارجع إلى البرنامج. احفظ البرنامج كبرنامج نصي تحت اسم BLSAVE.BAS مع اخلاء الشاشة.

٤ - انتقل إلى الدرس الخمسين للاستمرار في تسلسل التعلم.

الدرس العاشر

عبارتا CALL و CALLS

الوصف

عبارة CALL : تستدعى العبارة برنامجاً فرعياً مكتوباً بالبيسك السريع أو بأى لغة برمجة أخرى، وتكوينها هو كما يلي :

(قائمة البرنامج الفرعى) اسم البرنامج الفرعى CALL

اسم البرنامج الفرعى هو اسم البرنامج الفرعى المراد استدعاؤه، ويمكن أن يكتب الاسم بحد أقصى 40 خانة، وقائمة البرنامج الفرعى هي البيانات التى تمرر إلى مؤشرات البرنامج الفرعى، ويجب أن توضع أقواس حول هذا الجزء إلا إذا ما حذفت كلمة CALL وفى هذه الحالة يصبح تكوين العبارة على النحو التالى :

(قائمة البرنامج الفرعى) اسم البرنامج الفرعى

ويجب أن تكون القائمة من نفس النوع ولها نفس عدد مؤشرات البرنامج الفرعى،

عندما تستخدم عبارة CALL فى استدعاء برامج فرعية مكتوبة بلغات أخرى فتكون للقائمة تكوين مختلف كما يلي :

الكلمات الرئيسية BYVAL و SEG تعرف كيفية تمرير القائمة إلى البرنامج الفرعى، فتحدد BYVAL أن القائمة تمرر بالقيمة، وتحدد SEG أن القائمة تمرر كعنوان، ويجب أن تستخدم SEG بحذر عند تمرير المنظومات وذلك لأن بيسك السريع قد ينقل متغيرات فى الذاكرة قبل نقل التحكم إلى البرنامج الفرعى،

عندما تستخدم عبارة CALL بدون كلمة CALL فيجب أن توجد عبارة DECLARE لتمكن من التأكد من نوع القائمة.

عبارة CALLS : استخدام عبارة CALLS هو نفسه مثل استخدام عبارة CALL مع تمرير قائمة من نوع SEG إلى البرنامج المنادى، وتستخدم عبارة CALLS فى استدعاء برامج فرعية مكتوبة بلغات أخرى غير لغة بيسك السريع، ويتم تمرير القوائم كعناوين فقط، وتكوينها هو كما يلي :

قائمة البرنامج الفرعى) اسم برنامج فرعى CALLS

واسم البرنامج الفرعى هو اسم البرنامج الفرعى المراد استدعاؤه وتقع عليه نفس القيود التى تقع على اسم البرنامج الفرعى المستخدم فى عبارة CALL التى سبق ذكرها .

التطبيقات

تستخدم عبارات CALL و CALLS فى استدعاء برامج فرعية. وعبارات الاستدعاء CALL و CALLS تستخدم أساساً فى البرمجة بلغات مختلفة. وعند استدعاء برامج فرعية مكتوبة ببيسك السريع توفر عبارة CALL آلية مرنة لضمان المعلومات التى تمرر إلى الاجراء المندى عليه تكون من النوع الصحيح. وعند غياب كلمة CALL من عبارة CALL فتستخدم عبارة DECLARE لاجراء التأكد من نوع القائمة. وفيما يلي بعض الأمثلة :

مثال ١

```
CALL ShowMenu
SUB ShowMenu;
..
END SUB
```

مثال ٢

```
DECLARE SetUp (Q1 AS INTEGER)
SUB SetUp Q1
..
SUB SetUp(Q1 AS INTEGER)
..
END SUB
```

عملية تقليدية

يتعامل البرنامج المستخدم فى هذه العملية مع عبارة CALL التى تستدعى برامج فرعية مكتوبة ببيسك السريع. وقد أعد البرنامج فى الدرس الواحد والسبعين. ابدأ بتحميل ببيسك السريع.

١ - حمل البرنامج الموجود في الدرس الواحد والسبعين والمسمى LUBOUND.BAS.

٢ - عدل عبارات CALL إلى SUB كما هو موضح فيما يلي (كلمة CALL نفسها لم تستخدم):

```

File Edit View Search Run Debug Calls:
LUBOUND.BAS

'The following program demonstrates the use of the UBOUND and LBOUND
'statements. The program loads two sets of array values and
'finds the minimum and maximum values in those arrays.

DECLARE SUB FindMinMax (A%( ), MinVal!, MaxVal!)

Max = 15
DIM A%(Max)
GOTO Start

LoadArray:
FOR Cnt = 1 TO Max
    READ A%(Cnt)
NEXT
RETURN

Start:
CLS
GOSUB LoadArray
PRINT "First pass"
FindMinMax A%( ), MinVal, MaxVal
READ Max
REDIM A%(Max)
GOSUB LoadArray
PRINT "Second pass"
FindMinMax A%( ), MinVal, MaxVal

DATA 12,23,33,43,1,56,98,656,323,44,9,88,67,54,18
DATA 18
DATA 8,89,76,54,23,32,12,4,33,54
    
```

```

File Edit View Search Run Debug Calls:
LUBOUND.BAS:FindMinMax

SUB FindMinMax (A%( ), MinVal, MaxVal)
    MinVal = A%(1): MaxVal = A%(1)
    FOR Cnt = LBOUND(A%) + 1 TO UBOUND(A%)
        IF MinVal > A%(Cnt) THEN
            MinVal = A%(Cnt)
        END IF
        IF MaxVal < A%(Cnt) THEN
            MaxVal = A%(Cnt)
        END IF
    NEXT
    PRINT "Minimum value in array: "; MinVal, "Maximum value in array: "; MaxVal
END SUB

Immediate

Main: LUBOUND.BAS Context: Program not running 00001:001
    
```

- ٣ - نفذ البرنامج ولاحظ استخدام عبارة CALL في البرنامج، لاحظ غياب كلمة CALL من عبارة CALL وغياب الأقواس من حول القائمة.
- ٤ - ارجع إلى البرنامج مع اخلاء الشاشة دون أن تحفظ البرنامج.
- ٥ - انتقل إلى الدرس الحادي عشر للاستمرار في تسلسل التعلم.

الدرس الحادى عشر

عبارة CALL ABSOLUTE

الوصف

تنقل هذه العبارة تنفيذ البرنامج إلى برنامج فرعى مكتوب بلغة الآلة، وتكوينها هو كما يلي :

CALL ABSOLUTE (متغير صحيح، قائمة)

القائمة هي القائمة الاختيارية التي تمرر إلى البرنامج الفرعى المكتوب بلغة الآلة، والمتغير الصحيح هو فرع الجزء المكتوب بلغة الآلة والذي ينتقل إليه التحكم، ويحدد القطاع باستخدام عبارة DEF SEG. والقوائم التي تمرر إلى البرنامج الفرعى عبارة عن مشيرات قريبة أو فروع من داخل القطاع الحالى.

ملاحظة

يجب أن يحمل بيسك السريع مع المكتبة السريعة "QB.QLB" لكى يمكن لهذه العبارة أن تعمل. (اكتب QB.LQB عند ملقن DOS واضغط على مفتاح الادخال).

التطبيقات

هذه العبارة توجد فى بيسك السريع للحفاظ على بعض التوافقية مع صيغ بيسك القديمة. فعبارات CALL و CALLS والتي تستدعى برامج فرعية مكتوبة بلغات أخرى تمثل طريقة سهلة لاستدعاء برامج فرعية مكتوبة بلغة المجمع، والبرامج الفرعية المكتوبة بلغة المجمع التى سبق اعدادها باستخدام بيسك المطور يجب أن تعدل إذا ما كانت تقبل قوائم سلاسل وذلك لأن واصفات السلاسل يكون طولها 4 بايت فى بيسك السريع، ويوجد مثال لهذه العبارة فى جزء البرنامج التالى :

```
DIM FastPrint(7)
'Get the offset of the array where the assembly routine will be placed.
Ptr = VARPTR(FastPrint(1))
'Set the segment to the segment of the array.
DEF SEG = VARSEG(FastPrint(1))
'Poke the instructions in memory beginning with the offset of the array.
FOR i = 1 to 40
    READ Byte
    POKE (Ptr + i), Byte
NEXT
'Execute the call with a parameter T.
CALL ABSOLUTE(T,VARPTR(FastPrint(1)))
```


تظهر تعليمات المقطع المكتوب بلغة الآلة فى هذا البرنامج فى عبارة DATA كقيمة ممثلة
بالنظام السادس عشرى.

عملية تقليدية

حيث أنه لا يمكن افتراض أن القارئ ملّم بالبرمجة بلغة المجمع لا يكون للعملية التقليدية أى
معنى عملى فى هذا الدرس.

انتقل إلى الدرس الثانى عشر للاستمرار فى تسلسل التعلم.

الدرس الثانى عشر

عبارات CALL INTERRUPT و CALL86OLD

الوصف

تنفذ العبارتان وظيفة استدعاء نظام التشغيل DOS من داخل بيسك السريع، وتكوين كل منهما هو كما يلى :

```
CALL INT86OLD(intr, in_array, out_array)
CALL INT86XOLD(intr, in_array, out_array)
CALL INTERRUPT(intr, inregs, outregs)
CALL INTERRUPTX(intr, inregs, outregs)
```

لاحظ أن مؤشرات كل من الأربعة تكوينات هي نفس المؤشرات، يحدد جزء intr رقم ازعاج DOS، وجزئى in-array و out-array فى تكوينات INT86OLD و INT86XOLD هما منظومتان عدديتان صحيحتان فى نوعهما مع ثمانية عناصر لعبارة INT86OLD وعشرة عناصر لعبارة INT86XOLD. وعناصر المنظومة هي كما يلى :

in_array(1)	Register AX
in_array(2)	" BX
in_array(3)	" CX
in_array(4)	" DX
in_array(5)	" BP
in_array(6)	" SI
in_array(7)	" DI
in_array(8)	" Flags

كما أن المنظومة تشمل كذلك العنصرين التاليين فى INT86XOLD :

in_array(9)	" DI
in_array(10)	" FI

وتتبع المنظومة out-array نفس التكوين، وكل المسجلات باستثناء BP و DS تعدل بواسطة الاستدعاءات.

وفى تكوين CALL INTERRUPT تكون الاعداد الصحيحة والمنظومة على الصورة التالية :

```

TYPE  RegType
      AX AS INTEGER
      BX AS INTEGER
      CX AS INTEGER
      DX AS INTEGER
      BP AS INTEGER
      SI AS INTEGER
      DI AS INTEGER
      FLAGS AS INTEGER
      DS AS INTEGER
      ES AS INTEGER
END TYPE

```

ويوجد توضيح النوع فى ملف الشمول "Qb. Bi" وهو ضرورى لاستخدام وظائف الاستدعاء هذه.

يحتوى جزء in-array فى كل عبارات CALL السابقة على قيم مسجلات تستخدم فى تنفيذ ازعاج DOS ويحتوى جزء out-array على قيم المسجلات بعد حدوث الازعاج.

وتوزع دوال INT86OLD و INT86XOLD و INTERRUPT و INTERRUPTX كمكتبة سريعة مع المترجم عندما يتم تنفيذ بيسك السريع مع خيار سطر الأمر /lqb. ووظائف الازعاج هى وسائل برمجة مطورة تستخدم فى الحصول على معلومات من DOS وأداء ازعاجات DOS مباشرة. والإزعاجات interrupts هى أنشطة تزعج أى شئ آخر يجب أن يحدث لأداء وظيفة محددة من وظائف DOS. ويأخذ العديد من المقاطع مثل هذه الكتابة إلى الشاشة، مع اختيار حالة الرؤية video، وانتاج ملف ينفذ على أنه ازعاجات. إلا أن هذه تنقل إلى المستفيد وعلى هذا فلايحتاج المستفيد أن يهتم بها.

ملاحظة

يجب أن يحمل بيسك السريع مع المكتبة السريعة "QB. QLB" لكى يمكن لهذه العبارة أن تعمل. (اكتب QB\QB عند ملقن DOS واضغط على مفتاح الادخال).

التطبيقات

تصبح استدعاءات وظيفة DOS ضرورية عندما لا يمكن تنفيذ عملية معينة بوسيلة التطوير المستخدمة. احدى هذه المواقف يمكن أن يكون ايجاد معلومات عن النظام فى بداية البرنامج مثل

ما إذا كان هناك مطبع adaptor للرسومات الملونة أو لا أو ما هو حجم الذاكرة المتاحة. ويجب أن تستخدم ازماعات DOS إذا لم يكن هناك أى طريقة أخرى لأداء هذه الأنشطة فقط. وفيما يلي أمثلة لعبارات الازماعات CALL :

مثال ١

```
'Include the header file from QuickBASIC.
'$INCLUDE:'Qb.Bi'

DIM in_array(7), out_array(7)
in_array(1) = ..
in_array(3) = ..
in_array(4) = ..
intr = ..

CALL INT86OLD(intr,in_array(),out_array())
```

مثال ٢

```
DIM inregs AS RegType, outregs AS RegType
inregs.AX = ..: inregs.CX = ..: inregs.DX = ..
intr = ..

CALL INTERRUPT(intr,inregs,outregs)
```

وتقدم وظائف INT86OLD و INT86XOLD للحفاظ على التوافقية مع صيغ البيست الأخرى.

عملية تقليدية

هذه العملية توضح عبارة CALL INTERRUPT. ابدأ عند ملقن DOS.

١ - اكتب QBLQB واضغط على مفتاح الادخال. (ارجع إلى الملحق B للمزيد من المعلومات عن تحميل المكتبات السريعة).

٢ - اكتب البرنامج التالي :

- ٤ - ارجع إلى البرنامج مع اخلاء الشاشة دون أن تحفظ البرنامج.
- ٥ - انتقل إلى الدرس الرابع عشر للاستمرار في تسلسل التعلم.

الدرس الثالث عشر

دوال CDBL و CINT و CLNG و CSNG

الوصف

دالة CDBL : هذه الدالة تحول التعبير العددي إلى قيمة مزدوجة الدقة، وتكوينها هو كما يلي :

(تعبير عددي) CDBL

التعبير العددي هو القيمة التي تتحول إلى عدد له دقة مزدوجة، ولا يعنى ذلك أن النتيجة تكون أكثر دقة من العدد الأصلي بالرغم من وجود أرقام دقة أكثر، والدالة نفس التأثير مثل تحديد تعبير عددي بأنه متغير مزدوج الدقة.

دالة CINT : هذه الدالة تحول التعبير العددي إلى قيمة صحيحة عن طريق التقريب، وتكوينها هو كما يلي :

(تعبير عددي) CINT

التعبير العددي هو القيمة التي يحدث لها تقريب وتعود كقيمة عددية صحيحة، ويجب أن تقع القيمة العددية بين -32,768 و 32,767. وعندما لا يتحقق ذلك تظهر رسالة السريان الزائد، وعلى عكس دالتي FIX و INT اللتين تحذفان الكسر كاملاً فإن دالة CINT تقرب القيمة.

دالة CLNG : هذه الدالة تحول التعبير العددي إلى قيمة صحيحة طويلة، وتكوينها هو كما يلي :

(تعبير عددي) CLNG

تتحول قيمة التعبير العددي إلى عدد صحيح طويل (4 بايت)، وعندما لا تقع قيمة التعبير العددي في المدى من -2,147,483,648 إلى 2,147,483,647 فتظهر رسالة السريان الزائد.

دالة CSNG : هذه الدالة تحول التعبير العددي إلى قيمة فردية الدقة، وتكوينها هو كما يلي :

(تعبير عددي) CSNG

تحول القيمة العددية إلى عدد له دقة فردية، وهذا يشبه تحديد تعبير عددي لمتغير له دقة فردية، ويتم تقريب هذه القيمة إذا ما كانت هناك حاجة إلى ذلك.

التطبيقات

نوال CDBL و CINT و CLNG و CSNG هي جزء من ترسانة إجراءات تحويل النوع. وتستخدم هذه النوال في تحويل قيم عددية من نوع إلى آخر عندما تكون هناك حاجة إلى ذلك. وفيما يلي بعض الأمثلة :

مثال ١

```
DEFSNG q-t: DEFDBL l-p
Qrt = 32.01/.021
Press = CDBL(Qrt)
```

مثال ٢

```
DEFINT m-p: DEFLEN r-t
Mts = 32 * 1000
Rts = CLNG(Mts)
```

توضح الأمثلة استخدام عملية تحويل الأنواع العددية.

عملية تقليدية

توضح هذه العملية استخدام نوال CDBL و CINT و CLNG و CSNG. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :


```

File Edit View Search Run Debug Calls F1=Help
<Untitled>

'This program demonstrates the use of the CSNG, CINT, CDBL and CLNG
'functions. The program assigns values to variables defined to
'be of other types.

DEFINT I-L: DEFSNG M-P: DEFDBL Q-T
DEFLNG A-D

I = 128: L = 256
M = 22 / 7
Q = 22 / 7
A = 256000

CLS
PRINT "DEFINT vars. ": I, L
PRINT "DEFSNG vars. ": M
PRINT "DEFDBL vars. ": Q
PRINT "DEFLNG vars. ": A

I = CINT(A / 10)
A = CLNG(10! * 10)
M = CSNG(Q)
Q = CDBL(13 / 2)

PRINT "After CINT ": I
PRINT "After CLNG ": A
PRINT "After CSNG ": M
PRINT "After CDBL ": Q

Immediate

Main: <Untitled> Context: Program not running L 00006:02.1

```

٢ - نفذ البرنامج. لاحظ استخدام دوال CDBL و CINT و CLNG و CSNG في البرنامج.

```

DEFINT vars. 128 256
DEFSNG vars. 3.142857
DEFDBL vars. 3.142857074737549
DEFLNG vars. 256000
After CINT 25600
After CLNG 100
After CSNG 3.142857
After CDBL 6.5

Press any key to continue

```

٣ - ارجع إلى البرنامج مع إخلاء الشاشة دون أن تحفظ البرنامج.

٤ - انتقل إلى الدرس الثلاثين للاستمرار في تسلسل التعلم.

الدرس الرابع عشر

عبارة CHAIN

الوصف

تتسبب عبارة CHAIN فى تنفيذ برنامج بيسك سريع من برنامج بيسك سريع آخر. وتكوينها هو كما يلى :

CHAIN file spec

جزء file spec هو اسم ملف البرنامج المراد تنفيذه. ولا يعود التحكم إلى البرنامج المنادى بعد تنفيذ البرنامج المطلوب. والبرامج التى تنفذ فى بيئة تطوير بيسك السريع لا يمكنها أن تستدعى برامج بيسك قائمة بذاتها (.EXE). والبرامج التى تنفذ على أنها برامج قائمة بذاتها لا تستطيع أن تستدعى ملفات مصدر بيسك السريع. ويمكن أن تستخدم المتغيرات فى البرنامجين وتممر إلى البرنامج المنادى عليه من خلال مجموعة مشاركة COMMON فارغة.

التطبيقات

تستخدم عبارة CHAIN فى استدعاء برامج بيسك السريع من داخل البرنامج. والبرنامج المنادى عليه يمكن توصيله بالبرنامج الأسمى وحيث إن عبارة CHAIN لا تقبل أرقاماً للأسطر كجزء من مواصفة الملف فيجب أن تكتب عملية التسلسل بعناية فائقة لتجنب عمل الدورات اللانهائية. وفيما يلى بعض الأمثلة لعبارة CHAIN :

مثال ١

```
IF Proc > 2 THEN CHAIN "NewProc.Bas"
```

مثال ٢

```
SELECT CASE Choice$
CASE "E"
CHAIN "EditProc.Bas"
CASE "T"
CHAIN "Train.Bas"
END SELECT
```

عملية تقليدية

توضح العملية التالية استخدام عبارة CHAIN. ابدأ بتحميل ببسك السريع.

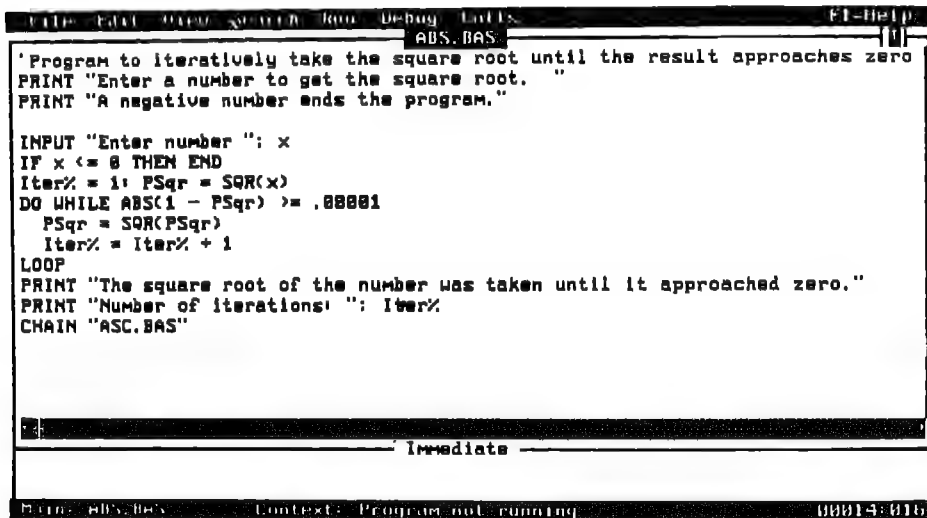
١ - حمل برنامج ABS.BAS وأضف السطر التالي في نهاية البرنامج :

```
CHAIN "ASC.BAS"
```

ملاحظة

إذا لم يشمل متغير PATH من DOS عندك دليل المصدر لببسك السريع أو إذا لم تكن تعمل ببسك السريع من دليل المصدر فيجب أن تقدم اسم المسار كجزء من اسم الملف مثلما يلي:

```
"\basic\qb.4\illqb.arc\Asc.Bas."
```



```

FILE EDIT VIEW SOURCE RUN DEBUG TOLLS
ABS.BAS
'Program to iteratively take the square root until the result approaches zero
PRINT "Enter a number to get the square root. "
PRINT "A negative number ends the program."

INPUT "Enter number ": x
IF x <= 0 THEN END
Iter% = 1: PSqr = SQR(x)
DO WHILE ABS(1 - PSqr) >= .00001
    PSqr = SQR(PSqr)
    Iter% = Iter% + 1
LOOP
PRINT "The square root of the number was taken until it approached zero."
PRINT "Number of iterations: "; Iter%
CHAIN "ASC.BAS"

Immediate

Date: 8/5/85 Context: Program not running 00014:016
    
```

٢ - نفذ البرنامج. بعد تنفيذ ببسك السريع لبرنامج ABS.BAS يظهر ملقن يسألك إذا ما كنت

تريد أن تحفظ الملفات ام لا. اكتب N ولاحظ أن البرنامج ينفذ الملف ASC.BAS.

٣ - ارجع إلى البرنامج مع اخلاء الشاشة دون أن تحفظ البرنامج.

٤ - انتقل إلى الدرس الثاني والعشرين للاستمرار في تسلسل التعلم.

الدرس الخامس عشر

دالة CHR\$

الوصف

تعيد دالة CHR\$ الرمز الذي تتفق قيمته في ASCII مع المؤشر الذي يمر، وتكوينها هو مايلي:

CHR\$(تعبير عددي)

وحيث إن الرمز الذي يعبر يكون من ضمن مجموعة رموز ASCII فدانماً ما يقع التعبير العددي للدالة بين 0 و 255.

التطبيقات

في البرنامج العينة الموجود في الدرس الثالث حدث استخدام لهذه الدالة في طباعة رموز شبه مرسومة من مجموعة ASCII المتسعة أثناء رسم المستطيل، وبعض الاستخدامات الأخرى تحدث لإنتاج سلاسل مكونة من رموز غير الرموز المعتاد استخدامها في الطباعة، وفيما يلي بعض الأمثلة.

```
OddStr$ = CHR$(13)+CHR$(10)
```

يصف هذا المثال عودة العربة وتسلسل تغذية السطر وعادة ما يستخدم كنهاية لعلامة سجل.

```
OddStr$ = CHR$(34)+  
"Embedded quotes! Otherwise not allowed!" + CHR$(34)
```

يبين هذا المثال كيفية اضافة علامتى تنصيص مزدوجتين إلى سلسلة.

عملية تقليدية

لرؤية الرموز المختلفة الموجودة في مجموعة ASCII وتوضيح دالة CHR\$ في بيسك السريع استمر كما يلي. ابدأ بتحميل بيسك السريع.

١ - اضغط على F6 للقفز إلى النافذة الفورية. ويمكنك من هذا المكان أن تكتب سطرًا من الشفرة وتنفذه على الفور. وتستطيع أن تحصل على معظم المعلومات عن النافذة الفورية من الملحق B.

٢ - اكتب ما يلي :

```
FOR i=1 TO 255 :PRINT i "=" CHR$(i) :NEXT
```

٣ - اضغط على مفتاح الإدخال لتنفيذ ذلك.

لاحظ أن بعض الرموز التي سبق طباعتها مبكراً أثناء تنفيذ البرنامج أظهرت أشياء غريبة على الشاشة. وهذا لأن هذه الرموز غير قابلة للطباعة. والرموز الوحيدة التي تترك على الشاشة هي التي تبدأ من (13) CHR\$ وما بعدها وذلك لأن (12) CHR\$ يخلي الشاشة وتطبع الرموز الأخرى منذ بداية الشاشة. يحتوى الملحق C على قائمة كاملة بكل الرموز الموجودة في مجموعة ASCII.

```
13 =
14 =N 15 =M 16 => 17 31 =8 =+ 19 =!! 28 =π 21 =§ 22 =- 23 =i 24 =† 25 =↓ 26 ==>
27 ==+ 28 = 29 30 = 32 = 33 =† 34 =" 35 == 36 =$ 37 =% 38 =& 39 =' 40 =(
41 =) 42 =* 43 =+ 44 =, 45 =- 46 =. 47 =/ 48 =0 49 =1 50 =2 51 =3 52 =4 53 =5
54 =6 55 =7 56 =8 57 =9 58 =: 59 =; 60 =< 61 == 62 => 63 =? 64 =@ 65 =A 66 =B
67 =C 68 =D 69 =E 70 =F 71 =G 72 =H 73 =I 74 =J 75 =K 76 =L 77 =M 78 =N 79 =O
80 =P 81 =Q 82 =R 83 =S 84 =T 85 =U 86 =V 87 =W 88 =X 89 =Y 90 =Z 91 =[ 92 =\
93 =^ 94 =_ 95 =` 96 =' 97 =a 98 =b 99 =c 100 =d 101 =e 102 =f 103 =g 104 =h
105 =i 106 =j 107 =k 108 =l 109 =m 110 =n 111 =o 112 =p 113 =q 114 =r 115 =s
116 =t 117 =u 118 =v 119 =w 120 =x 121 =y 122 =z 123 =< 124 =| 125 => 126 =~
127 =^ 128 =_ 129 =u 130 =e 131 =A 132 =a 133 =A 134 =a 135 =g 136 =B 137 =b
138 =B 139 =b 140 =f 141 =1 142 =A 143 =A 144 =E 145 =e 146 =F 147 =B 148 =b
149 =B 150 =Q 151 =U 152 =y 153 =O 154 =U 155 =C 156 =c 157 =F 158 =R 159 =f
160 =B 161 =i 162 =B 163 =U 164 =F 165 =N 166 =N 167 =a 168 =L 169 =r 170 =r
171 =B 172 =B 173 =i 174 =C 175 => 176 =B 177 =B 178 =B 179 =| 180 =| 181 =|
182 =| 183 =π 184 = 185 =| 186 =| 187 =π 188 = 189 = 190 = 191 = 192 =L
193 = 194 = 195 = 196 = 197 = 198 = 199 = 200 = 201 = 202 = 203 =
204 = 205 = 206 = 207 = 208 = 209 = 210 = 211 = 212 = 213 = 214 =
215 = 216 = 217 = 218 = 219 = 220 = 221 = 222 = 223 = 224 = 225 =
226 = 227 = 228 = 229 = 230 = 231 = 232 = 233 = 234 = 235 = 236 =
237 = 238 = 239 = 240 = 241 = 242 = 243 = 244 = 245 = 246 = 247 =
248 = 249 = 250 = 251 = 252 = 253 = 254 = 255 =

Press any key to continue
```

٣ - اضغط على أي مفتاح للعودة إلى النافذة الفورية.

٤ - اضغط على F6 لترك النافذة الفورية والعودة إلى نافذة الرؤية.

٥ - انتقل إلى الدرس المائة والثالث والأربعين للاستمرار في تسلسل التعلم.

الدرس السادس عشر

عبارة CIRCLE

الوصف

ترسم عبارة CIRCLE دائرة على الشاشة وذلك طبقاً لقيمة مركز ونصف قطر معينين، وتكوينها هو كما يلي :

`CIRCLE STEP (x,y),radius,color,start,end,aspect`

يحدد جزء STEP أن إحداثيات x و y نسبية للوضع الحالي وهو جزء اختياري، وجزء (x, y) هي إحداثيات الشاشة لمركز الدائرة، وجزء radius هو نصف القطر المستخدم في رسم الدائرة، وجزء start وجزء end يستخدمان في رسم جزء من دائرة أو رسم أقواس، وتقع قيم start و end في مدى يتراوح من -2π إلى 2π (حيث $\pi = 3.141593$) وقيمهما التقليدية هي 0 لجزء start و 2π لجزء end، وعندما تكون قيمة أي منهما سالبة فيرسم القوس مع اعتبار أن القيمة موجبة، وجزء aspect هو نسبة y دائرياً إلى x دائرياً وتحسب قيمته التقديرية كما يلي :

$$4 * (y \text{ pixels} / x \text{ pixels}) / 3$$

حيث نقاط الرسم الرأسية y pixels ونقاط الرسم الأفقية x pixels هما قيمة الثبات للشاشة ويعتمدان على حالة الشاشة، وعندما يكون aspect أقل من 1 فإن radius يكون نصف قطر x أما عندما يكون أكبر من 1 فيكون radius هو نصف قطر y.

التطبيقات

عبارة CIRCLE هي وسيلة أخرى متعددة الجوانب في تطبيقات الرسومات، وتقوم العبارة برسم الدائرة كلها أو جزءاً منها أو ترسم قطعاً ناقصاً أو قطاعاً منه وذلك طبقاً للمؤشرات، ومقدرة استخدام إحداثيات مطلقة أو نسبية (باستخدام STEP) هي من المميزات الإضافية، وبعد رسم الدائرة تشير عبارة CIRCLE إلى مركزها أي أنك إذا رسمت دائرتين مستخدماً نفس المؤشرات فإنهما يرسمان في نفس المكان إلا إذا حددت إحداثيات نسبية، وفيما يلي أمثلة لعبارة CIRCLE :

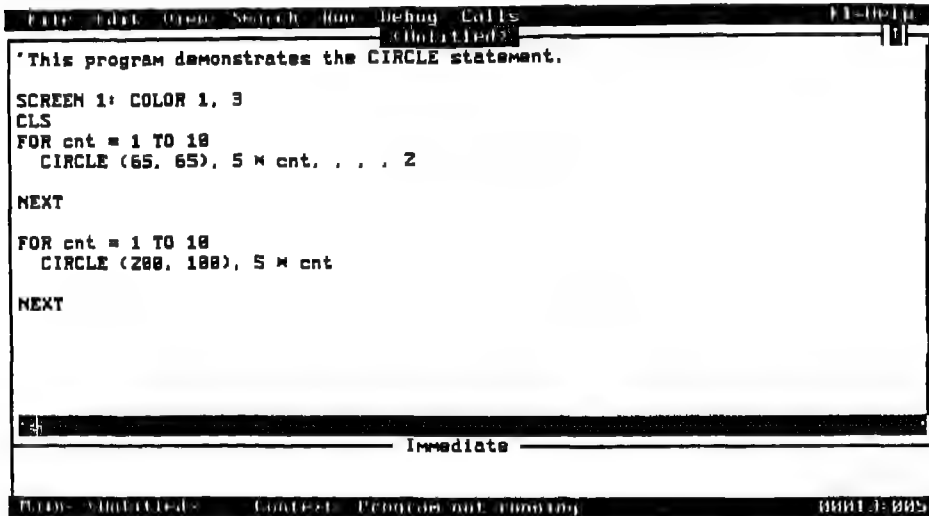
CIRCLE

```
CIRCLE (180,200),50
CIRCLE (200,100),100,,1,2
CIRCLE (50,200),50,,.05,-.1
```

عملية تقليدية

توضح هذه العملية استخدام عبارة CIRCLE في صورة مبسطة. استمر إذا ما كان لديك نظام شاشة رسومات ملونة تدعم ذلك فقط. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :



```
File Edit View Search Run Debug Calls
C:\Program Files\Microsoft\VB\VB60\VB60.exe
'This program demonstrates the CIRCLE statement.
SCREEN 1: COLOR 1, 3
CLS
FOR cnt = 1 TO 10
  CIRCLE (65, 65), 5 * cnt, , , 2
NEXT
FOR cnt = 1 TO 10
  CIRCLE (280, 180), 5 * cnt
NEXT
Immediate
00010: 0005
```

- ٢ - نفذ البرنامج. لاحظ استخدام عبارة CIRCLE في البرنامج. لاحظ كذلك استخدام مؤشر aspect وكيفية تأثيره على المخرجات.
- ٣ - ارجع إلى البرنامج مع اخلاء الشاشة نون أن تحفظ البرنامج.
- ٤ - انتقل إلى الدرس المائة وستة للاستمرار في تسلسل التعلم.

الدرس السابع عشر

عبارة CLEAR

الوصف

تغلق عبارة CLEAR كل الملفات المفتوحة وتضع قيماً ابتدائية لمتغير البرنامج وتخلي الرصة وتغير حجمها (اختيارياً)، وتكوينها هو كما يلي :

```
CLEAR ..stack
```

يلزم وجود الفاصلتين للتوافقية مع صيغ بيسك القديمة ويعرف جزء stack موقع الرصة الاضافى اللازم بعد الرصة الموجودة حالياً، وإيجازاً لعبارة CLEAR تؤدي الأنشطة التالية :

- تغلق كل الملفات المفتوحة وتزيل مكان الذاكرة الاحتياطية.
- تزيل كل متغيرات COMMON.
- تضع أصفاراً كقيم ابتدائية للمتغيرات العددية وتضع فراغات كقيم ابتدائية لمتغيرات السلاسل.
- تخلي الرصة وتغير (اختيارياً) من حجمها.

التطبيقات

يمكن أن تستخدم عبارة CLEAR فى انتاج مكان أكبر للرصة للبرامج التى لها برامج فرعية متداخلة بعمق أو التى لها اجراءات اعادة ذاتية عديدة، وفيما يلي أمثلة لاستخدام عبارة CLEAR.

مثال ١

```
CLEAR ..
```

مثال ٢

```
CLEAR ..,4048
```


يزيل المثال الأول المتغيرات ويفلق الملفات فقط، أما المثال الثاني فيؤدي هذه الأنشطة بالإضافة إلى أنه يزيد من حجم الرصة بمقدار 4048 بايت. ويجب ألا تستخدم عبارة CLEAR من داخل برنامج فرعى وذلك بسبب اخلاء الرصة وفقدان عنوان العودة إلى عبارة النداء.

عملية تقليدية

هذه العملية تستخدم برنامجاً تم اعداده في الدرس المائة والسابع والثلاثين وتم تنقيحه في الدرس الخمسين. وهذا البرنامج معدل هنا لتوضيح استخدام عبارة CLEAR. ابدأ بتحميل ببسك السريع.

١ - حمل البرنامج STA_DYN.BAS وأضف آخر عبارتين كما هو موضح فيما يلي :

```

* This program demonstrates the FRE function.
* The program declares arrays and deallocates them.
* The memory available before and after is displayed on the screen.

REM $STATIC
DIM W(100, 100)
REM $DYNAMIC
DIM Q(10000)
CLS
PRINT "Amount of free memory ";
PRINT FRE(-1); "/"; FRE(-2); "/"; FRE(10); "/"; FRE("Junk")

REDIM Q(1000)
PRINT "After redimensioning Q ";
PRINT FRE(-1); "/"; FRE(-2); "/"; FRE(10); "/"; FRE("Junk")

ERASE Q
PRINT "After erasing Q ";
PRINT FRE(-1); "/"; FRE(-2); "/"; FRE(10); "/"; FRE("Junk")

CLEAR , , 4048

PRINT FRE(-1); "/"; FRE(-2); "/"; FRE(10); "/"; FRE("Junk")

```

٢ - نفذ البرنامج. وتكون المخرجات كما يلي. لاحظ ثالث قيمة في آخر سطرين والتي تعكس الانكماش في موقع الرصة.

```
Amount of free memory 262240 / 1162 / 49808 / 49808
After redimensioning Q 298240 / 1162 / 49808 / 49808
After erasing Q 302256 / 1162 / 49808 / 49808
340272 / 3962 / 47008 / 47008
```

Press any key to continue

٣ - ارجع إلى البرنامج مع اخلاء الشاشة دون أن تحفظ البرنامج.

٤ - انتقل إلى الدرس الثامن للاستمرار في تسلسل التعلم.

الدرس الثامن عشر

عبارة CLOSE

الوصف

تغلق عبارة CLOSE المدخلات الملف أو المخرجات إلى ملف أو إلى أحد الوحدات. وتكوينها هو كما يلي :

CLOSE #filenum

جزء filenum هو رقم الملف المحدد في عبارة OPEN. ويمكن للعبارة أن تغلق (اختيارياً) ملفات عديدة مرة واحدة عن طريق كتابة أرقام ملفات إضافية مفصولة عن بعضها البعض بواسطة فواصل.

عندما تستخدم هذه العبارة بدون رقم للملف فإنها تغلق كل الملفات وكل الوحدات. والملف المصاحب لرقم الملف يتوقف عن كونه متاحاً بعد تنفيذ عبارة CLOSE المستخدمة لهذا الرقم كمؤشر لها. وعند ذلك يصبح هذا الرقم متاحاً لإعادة التحديد لأي ملف آخر. وعند استخدام عبارة CLOSE مع ملف تتابعي أو مع وحدة تتابعية فتكتب العبارة آخر ذاكرة احتياطية وتجعل موقع الذاكرة الاحتياطية متاحاً دائماً للبرنامج.

نصيحة : هناك عبارات أخرى لإغلاق كل الملفات ذاتياً وهي CLEAR و END و RESET و RUN و SYSTEM.

التطبيقات

هذه العبارة ضرورية في تشغيل الملفات وذلك لضمان اتمام العمل بالترتيب المطلوب وسلامة البيانات المخزنة والمستعادة. وفيما يلي أمثلة لهذه العبارة :

مثال ١

```
OPEN "Sample.Dat" FOR RANDOM AS #1
...
CLOSE #1
```

مثال ٢

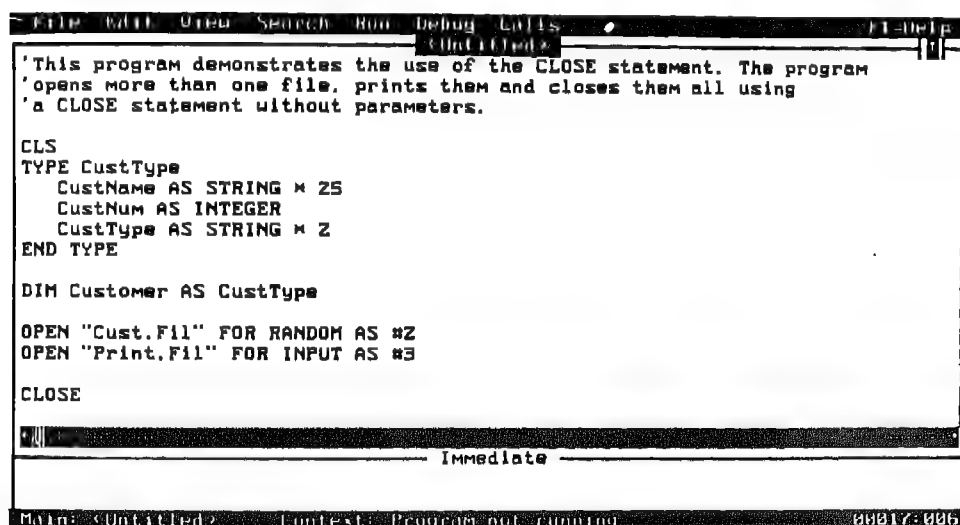
```
OPEN "File.001" FOR INPUT AS #1
OPEN "File.002" FOR OUTPUT AS #2
OPEN "File.003" FOR RANDOM AS #3
OPEN "COM2:1200,7,E,1,ASC" AS #11
CLOSE
```

يوضح المثال السابق استخدام هذه العبارة بدون مؤشرات وذلك في اغلاق كل الملفات والوحدات المفتوحة.

عملية تقليدية

توضح العملية استخدام هذه العبارة. ابدأ بتحميل ببسك السريع.

١ - اكتب البرنامج التالي :



```
'This program demonstrates the use of the CLOSE statement. The program
'opens more than one file, prints them and closes them all using
'a CLOSE statement without parameters.

CLS
TYPE CustType
  CustName AS STRING * 25
  CustNum AS INTEGER
  CustType AS STRING * 2
END TYPE

DIM Customer AS CustType

OPEN "Cust.Fil" FOR RANDOM AS #2
OPEN "Print.Fil" FOR INPUT AS #3

CLOSE

Immediate
```

٢ - نفذ البرنامج. لاحظ استخدام عبارة CLOSE في البرنامج.

٣ - ارجع إلى البرنامج ثم اختر New ولا تحفظ هذا البرنامج.

٤ - انتقل إلى الدرس الثامن والثمانين للاستمرار في تسلسل التعلم.

الدرس التاسع عشر

عبارة CLS

الوصف

تخلى هذه العبارة الشاشة وتترك نقطة البداية في الركن العلوي الأيسر للشاشة. وتكوينها هو كما يلي :

CLS 0|1|2

الجزء CLS هو إحدى كلمات البيسك المحجوزة، ويمكن أن يتبعه اختيارياً أى من القيم التالية مفصولة عن بعضها البعض بواسطة |. وعندما يستخدم بدون مؤشرات فإنه يخلى بوابة المشاهدة النشطة في الحالة النصية أو في حالة الرسومات، وفي الحالة النصية يقوم بإعادة نشاط عرض مفتاح الوظيفة الموجود في قاعدة الشاشة إذا ما كان في الوضع المفتوح. الجزء 0 هو مؤشر اختياري، ويقوم بإخلاء الشاشة من كل النصوص والرسومات. الجزء 1 هو مؤشر اختياري يقوم بإخلاء بوابة مشاهدة الرسومات فقط إذا ما نفذت عبارة VIEW، وإلا فإنه يخلى الشاشة كلها. أما الجزء 2 فهو مؤشر اختياري يخلى بوابة مشاهدة النصوص فقط مع ترك السطر الموجود في القاعدة دون أى تغيير.

التطبيقات

تقدم عبارة CLS طريقة فعالة لإخلاء الشاشة وتستخدم كلما دعت الحاجة إلى شاشة جديدة أثناء تنفيذ البرنامج. ويكون هذا مفيداً قبل عرض مخرجات البرنامج أو أخذ مدخلات له. وفيما يلي أمثلة لذلك :

مثال ١

```
CLS: INPUT "Enter filename ", F$
```

مثال ٢

```
CLS
LOCATE 23,1: PRINT "E)dit / B)rowse / P)rint / Q)uit "
Choice$ = INKEY$
```

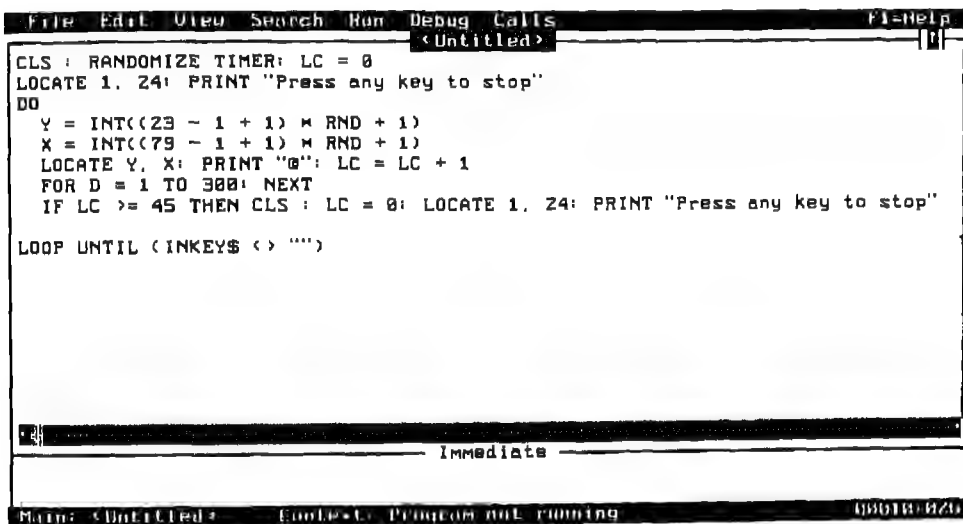
مثال ٢

```
GOSUB UpdateRecords
CLS
GOSUB DisplayReports
```

عملية تقليدية

تقوم في هذه العملية بادخال وتنفيذ برنامج يوضح استخدام هذه العبارة، ابدأ بتحميل
بيسك السريع.

١ - اكتب البرنامج التالي :

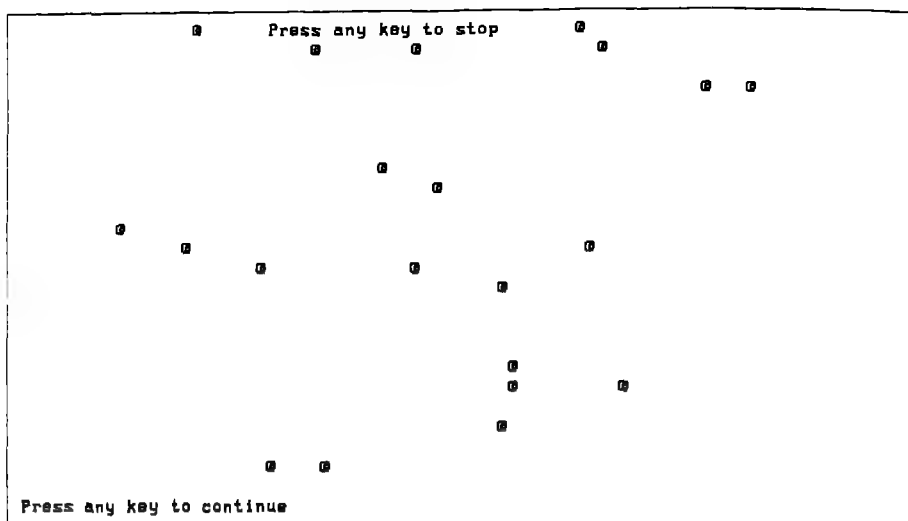


```
File Edit View Search Run Debug Calls F1=Help
<Untitled>
CLS : RANDOMIZE TIMER: LC = 0
LOCATE 1, 24: PRINT "Press any key to stop"
DO
Y = INT((23 - 1 + 1) * RND + 1)
X = INT((79 - 1 + 1) * RND + 1)
LOCATE Y, X: PRINT "a": LC = LC + 1
FOR D = 1 TO 300: NEXT
IF LC >= 45 THEN CLS : LC = 0: LOCATE 1, 24: PRINT "Press any key to stop"
LOOP UNTIL (INKEY$ <> "")

Immediate

Name: <Untitled> 640x480 Program not running
```

٢ - اضغط على Shift-F5 للتنفيذ، لاحظ استخدام عبارة CLS في اخلاء الشاشة بعد طباعة
45 رمزاً عليها، اضغط على أى مفتاح لإيقاف البرنامج.



٣ - اضغط على أى مفتاح للعودة إلى قائمة البرنامج.

٤ - اضغط على Alt-F ثم اضغط على مفتاح الإدخال واكتب N لإخلاء الشاشة.

٥ - انتقل إلى الدرس الخامس والخمسين للاستمرار فى تسلسل التعلم.

الدرس العشرون

عبارة COLOR

الوصف

تحدد هذه العبارة ألوان العرض، وتكوينها هو كما يلي :

```
COLOR foreground, background, border
COLOR background, palette
COLOR foreground, background
COLOR foreground
```

يستخدم التكوين الأول مع حالة الشاشة 0. وفي حالة الشاشة هذه يمكن اختيار أمامية الشاشة وخلفيتها وحدودها، جزء foreground عبارة عن رقم صحيح يتراوح من 0 إلى 31 ويضع اللون للرموز والأشياء الموجودة على الشاشة، وبعد الرقم 15 توضع الألوان، وجزء back-ground عبارة عن رقم صحيح يتراوح من 0 إلى 7 ويضع اللون لخلفية الشاشة، ومن غير المسموح به أن توضع الألوان في الخلفية، وجزء border عبارة عن رقم صحيح يتراوح من 0 إلى 15 ويضع حدود الشاشة، ولا يدعم أى من EGA أو VGA أو CGA حدود الشاشة.

ويستخدم التكوين الثانى مع حالة الشاشة 1 والأجزاء عبارة عن أرقام صحيحة تقع فى المدى الذى سبق ذكره، وفى هذه الحالة لا يمكن اختيار لون للأمامية إلا أنه يمكن اختيار لون للخلفية يتراوح من 0 إلى 7، وجزء المجموعة palette عبارة عن رقم فردى أو زوجى تتراوح قيمته من 0 إلى 255 ويختار مجموعة الألوان التى تعرض مع رقم معين للون.

ويستخدم التكوين الثالث فى حالات الشاشة من 7 إلى 10 وجزئى أمامية والخلفية يكونان تعبيرين صحيحين تتراوح قيمتهما فى المدى الذى سبق ذكره، وفى هذه الحالة لا يمكنك أن تحدد حدود الشاشة، وأمامية الشاشة تكون رقم الخاصية أما الخلفية فتكون رقماً لأحد الألوان.

ويستخدم التكوين الرابع فى حالات الشاشة 12 و 13، وتكون الأمامية عبارة عن تعبير صحيح كما سبق ذكره، وتحدد الخاصية لأمامية الرسومات.

التطبيقات

تستخدم عبارة COLOR مع عبارة SCREEN في تحديد الألوان المستخدمة أثناء ظهور المخرجات على الشاشة. والاختيار المناسب لألوان الشاشة لإحدى حالاتها يكون مهماً إذا ما كان البرنامج متداخلاً أو مستخدماً لرسومات وكانت عروض الشاشة مهمة في أحد التطبيقات. وفيما يلي أمثلة لهذه العبارة.

مثال ١

```
SCREEN 1
COLOR 1,2
CIRCLE (10,20),40
```

مثال ٢

```
COLOR 2,1,1
LINE STEP(100,150)-(50,50)
```

عملية تقليدية

هذه العملية توضح عبارة COLOR في صورة مبسطة، وذلك مع رسومات تظهر على شاشة أحادية اللون. حمل ببسك السريع أولاً.

١ - اختر Open وحمل البرنامج الذي تم اعداده في الدرس المائة وتسعة وهو PRINT.BAS.
٢ - نفذ البرنامج. لاحظ المخرجات الموجودة على الشاشة واستخدام عبارة COLOR في البرنامج.

٣ - ارجع إلى البرنامج مع اخلاء الشاشة بون أن تحفظ البرنامج.

٤ - انتقل إلى الدرس الخامس والعشرين للاستمرار في تسلسل التعلم.

الدرس الحادى والعشرون

دالة COMMAND\$

الوصف

تعيد دالة COMMAND\$ مؤشرات سطر الأمر التى تقدم عند استدعاء البرنامج. وتكوينها هو ما يلى :

COMMAND\$

تعيد الدالة سلسلة وهى السلسلة التى سبق طباعتها بعد اسم البرنامج عندما تم استدعاء البرنامج. وتحذف الفراغات الزائدة فى السلسلة وتحول حروفها إلى الحروف الكبيرة.

وتعيد دالة COMMAND\$ مؤشرات سطر الأوامر التى تقدم عند ملقن DOS عندما ينفذ برنامجاً قائماً بذاته. كما أنه من الممكن كذلك توفير مؤشر سطر الأمر عندما ينفذ البرنامج من بيسك السريع عن طريق استخدام خيار /cmd عند استدعاء بيسك السريع أو عند اختيار COMMAND\$ من قائمة Run.

التطبيقات

تستخدم دالة COMMAND\$ فى معرفة موقع مؤشرات سطر الأمر. وحيث إن الدالة تعيد سلسلة فقط وليس قائمة بمحتويات السلسلة لسطر الأمر فيكون من وظيفة المبرمج أن يفحص عناصر السلسلة ويعرف المؤشرات. وفيما يلى أمثلة لدالة COMMAND\$.

مثال ١

```
CmdParm$ = COMMAND$
SELECT CASE CmdParm$
CASE IS "MONO"
GOSUB SetMono
CASE IS "COLOR"
GOSUB SetColor
END SELECT
```

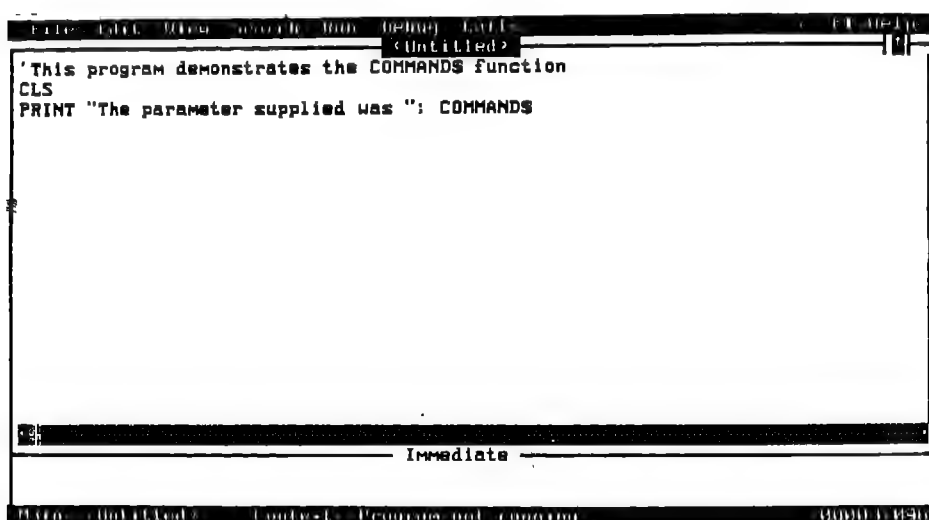
مثال ٢

```
DEFSTR A-E
Cparm = COMMAND$
IF Cparm = "" THEN
    GOSUB BusinessAsUsual
ELSE
    GOSUB ParseCommand
END IF
```

عملية تقليدية

يقبل البرنامج في هذه العملية مؤشراً واحداً لسطر أمر ويطبّع المؤشر المقدم. ابدأ بتحميل
بيسك السريع.

١ - اكتب البرنامج التالي :



٢ - انتقل إلى قائمة Run واختر COMMAND\$ واكتب dummy واضغط على مفتاح
الادخال ونفذ البرنامج. تظهر المخرجات على الصورة التالية :

The parameter supplied was DUMMY

Press any key to continue

- ٣ - لاحظ استخدام COMMAND\$ في البرنامج، قم بإخلاء الشاشة دون أن تحفظ البرنامج.
- ٤ - انتقل إلى الدرس السابع والثلاثين للاستمرار في تسلسل التعلم.

الدرس الثانى والعشرون

عبارة COMMON

الوصف

توضح عبارة المشاركة COMMON المتغيرات على أنها شاملة بحيث يمكن استخدامها بالمشاركة بين البرامج الفرعية والدوال والبرامج المتصلة ببعضها البعض، وتكوين هذه العبارة هو كما يلي :

```
COMMON SHARED /block name/ variable list
```

جزء SHARED اختياري ويلغى الحاجة إلى وجود عبارة SHARED داخل البرامج الفرعية والدوال لكي تتشارك في المتغيرات الشاملة، وجزء name/ block هو معرف مجموعة المشاركة، وهو معرف بيسك سريع صحيح، لا يزيد طوله عن 40 خانة، ويعرف اسم المجموعة مجموعة من المتغيرات كمجموعة يمكن اقتسامها أو المشاركة فيها كمجموعة. وعندما يحذف اسم المجموعة فتكون مجموعة المشاركة عبارة عن مجموعة فارغة وإلا فإنها تكون مجموعة مسماة. وعندما يحدث توصيل برامج ببعضها فتستخدم مجموعة مشاركة فارغة نظراً لأن مجموعة المشاركة المسماة لا تستخدم مع البرامج المتصلة ببعضها البعض، ويكون لقائمة المتغيرات الشكل التالي :

```
var1 AS type, var2 AS type,...
```

var1 و var2 هي أسماء متغيرات صحيحة في بيسك السريع ويحدد نوع AS نوع البيانات التي تنتمي إليها المتغيرات، ويمكن أن يكون النوع من أى نوع من الأنواع البسيطة أو التي يعدها المستفيد، ويمكن توضيح متغيرات المنظومات باستخدام أقواس فارغة، ويجب تحديد أبعاد المنظومات الاستاتيكية وذلك باستخدام ثوابت عددية في عبارة DIM وذلك قبل ظهور عبارة المشاركة أما المنظومات الديناميكية فيجب أن توضح في تسلسل عبارة DIM أو عبارة REDIM قبل استخدامها.

ويجب أن تظهر عبارة المشاركة قبل أى عبارات تنفيذ فى البرنامج وذلك نظراً لأن عبارة المشاركة تحدد مخازن للمتغيرات فى مكان معين فى الذاكرة. ويكون نوع وترتيب المتغيرات الموجودة فى مجموعة المشاركة أكثر أهمية من أسماء المتغيرات نفسها. لاحظ المثال التالى :

```
'Program A
COMMON Cnt1, Cnt2, Cnt3
Cnt1 = 21: Cnt2 = 0: Cnt3 = 100
..

'Program B
COMMON Cnt2, Cnt3, Cnt1
..
```

قيم $Cnt1=100$ و $Cnt2=21$ و $Cnt3=0$ فى البرنامج وذلك لأن ترتيب ظهورها كان موضحاً فى عبارة المشاركة.

ومن الممكن استخدام مجموعة مشاركة أكبر من مجموعة أخرى إذا ما كانت المشاركة المستخدمة من النوع الفارغ وليس النوع المسمى ولا يمكن إعادة تعريف مجموعة مشاركة لحجم أكبر عندما يقتسمها أحد البرامج مع برنامج فرعى من البرامج الموجودة فى المكتبة.

التطبيقات

عبارة المشاركة هى وسيلة قوية جداً للمشاركة فى استخدام المتغيرات بواسطة برامج فرعية وبرامج متصلة ببعضها البعض. ومع البرامج المتصلة ببعضها البعض تكون مجموعة المشاركة الفارغة هى الطريقة الوحيدة للمشاركة فى استخدام المتغيرات بين البرامج. وتقدم مجموعة المشاركة المسماة طريقة جيدة للمشاركة فى استخدام مجموعة محددة من المتغيرات بواسطة برامج فرعية. وفيما يلى أمثلة لاستخدام عبارة COMMON :

```
'Module 1
COMMON /QGroup/ Q, R, S, T
COMMON /RGroup/ R1, R2, R3
..

'module 2
COMMON /QGroup/ Q, R, S, T
..
```

يوضح المثال السابق كيف يمكن اقتسام استخدام مجموعة مسماة بحيث إنه يمكن الاتصال بمجموعة المتغيرات المحددة. وفى الدرس الثانى، المجموعة المسماة RGroup ليست متاحة.

```
'Module 1
'$INCLUDE: 'ComBlk.001'
..

'module 2
'$INCLUDE: 'ComBlk.001'
..

'ComBlk.001
COMMON SHARED /Pass1/ Lrt, Trt, Yrt
COMMON NewTrt, NewTrt, NewYrt
..
```

يوضح المثال السابق كيف يمكن وضع كل عبارات المشاركة في ملف شامل مع استخدامها في الأجزاء المختلفة. ويمكن أن يقلل ذلك بشدة من عبارات المشاركة COMMON غير المتوافقة مع بعضها البعض.

عندما تختلف مجموعات المشاركة في حجمها فتوضع قيم ابتدائية أصفراً للمتغيرات العددية وفراغات لمتغيرات السلسلة وذلك لكل المتغيرات الموجودة في مجموعة المشاركة الأكبر وغير الموجودة كجزء في مجموعة المشاركة الأصغر.

عملية تقليدية

هذه العملية هي توضيح بسيط لعبارة المشاركة. ويوضح البرنامج أهمية ترتيب ونوع المتغيرات عن أسمائها في عبارة المشاركة. ويتم إعداد البرنامج كعدة أجزاء منفصلة. ابدأ بتحميل بيسك السريع.

ملاحظة

إذا كانت هناك صعوبة في هذه العملية في إنتاج الملفات واختيار الخيارات فيجب أن ترجع إلى الملحق B (لإستخدام صناديق الحوار).

١ - اختر خيار Create من قائمة File واكتب COMMON.BAS واضغط على Tab مرتين ثم اضغط على مفتاح الإدخال. اكتب أسطر البرنامج التالية :

```

File Edit View Search Run Debug Calls :
COMMON.BAS
DECLARE SUB Prt ( )
'This is module 1
'$INCLUDE: 'ComBlk.001'
Prt1 = 10: Prt2 = 20: Prt3 = 30: Prt4! = .0004
CLS
PRINT "From Module 1:"
PRINT "Prt1 = "; Prt1: "Prt2 = "; Prt2: "Prt3 = "; Prt3: "Prt4! = "; Prt4!
CALL Prt

Immediate

Main: COMMON.BAS Context: Program not running 00000:001

```

٢ - اختر Create من قائمة File واكتب COMBLK.001. حدد أن نوع الملف هو Include واضغط على مفتاح الإدخال ثم اكتب أسطر البرنامج التالية :

```

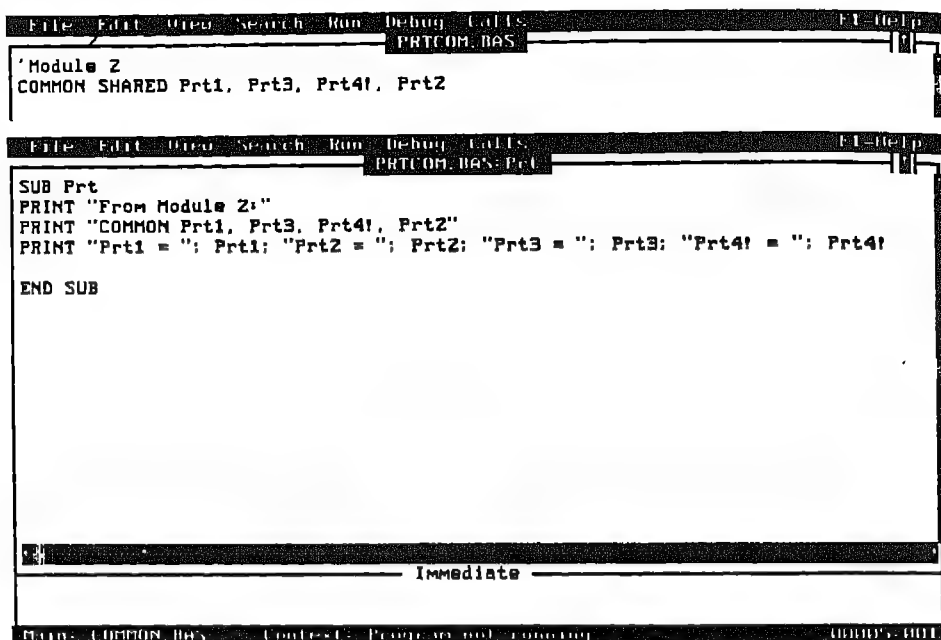
File Edit View Search Run Debug Calls :
COMBLK.001
'This file is the COMMON definition include file
COMMON SHARED Prt1, Prt2, Prt3, Prt4!

Immediate

Main: COMMON.BAS Context: Program not running 00001:001

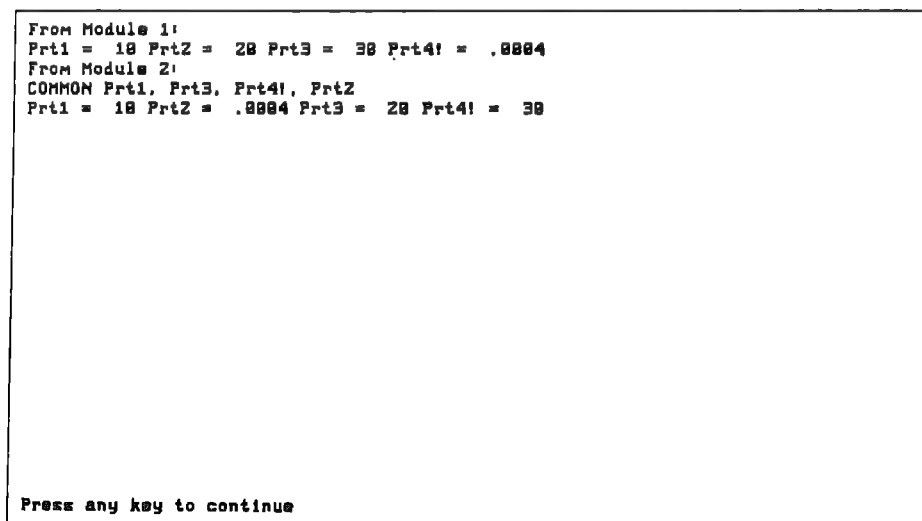
```

٣ - اكتب جزءاً آخر اسمه PRTCOM.BAS واكتب أسطر البرنامج التالية :



٤ - اختر Save All من قائمة File واحفظ كل الأجزاء التي سبق لك كتابتها.

٥ - اختر Set Main Module من قائمة Run واختر COMMON.BAS كجزء رئيس ونفذه.
تشبه المخرجات ما يلي :



لاحظ استخدام أمر \$INCLUDE فى الجزء الرئيسى والترتيب المختلف للمتغيرات فى الجزء PRTCOM.BAS ونتائج ذلك. لاحظ كذلك استخدام جزء SHARED فى عبارة المشاركة فى الملف المشمول وجزء PRTCOM. BAS. وفيما يلى بعض الأشياء التى تستحق أن تجربها :

- احذف جزء SHARED ونفذ البرنامج.
- أعد ترتيب المتغيرات ونفذ البرنامج.
- حاول أن تصل الجزء PRTCom. Bas.
- ٦ - ارجع إلى البرنامج مع اخلاء الشاشة دون أن تحفظ البرنامج.
- ٧ - انتقل إلى الدرس المائة والثانى والعشرين للاستمرار فى تسلسل التعلم.

الدرس الثالث والعشرون

الثابت CONST

الوصف

الثابت Constant عبارة عن قيمة لا تتغير أثناء تنفيذ البرنامج. ولتوضيح أن أحد المعارف ثابتاً يستخدم فعل CONST. انظر توضيحات الثوابت الموجودة في البرنامج BOX.BAS المقدم في الدرس الثالث. الثوابت الموجودة في هذا الدرس تسمى ثوابت رمزية symbolic constants. أما أنواع الثوابت الأخرى فتسمى ثوابت حرفية literal constants. وفيما يلي بعض الثوابت الرمزية:

```
CONST Pi = 3.1429, FICA = .0751
CONST CarriageRet = 24, EndOfFile = 26
CONST Title$ = "Sir/Madam", Address$ = "Dear " + Title$
CONST State$ = "Texas", City$ = "Dallas"
```

وفيما يلي بعض الثوابت الحرفية:

```
"Thou shalt not."
93000.23
44422
12000
"Today's Appointments: "
```

لاحظ أن الكلمة الرئيسية CONST من بيك السريع لم تستخدم في هذه الأمثلة. ويبين الجدول التالي أنواع الثوابت المختلفة المسموح باستخدامها في بيك السريع:

النوع	الوصف	مثال
Integer (Decimal)	قيمة صحيحة تقع فى المدى من -32,768 إلى 32,767 مع وجود اشارة اختيارية تسبقها.	+255,32, -768
Integer (Hexadecimal)	قيمة سادسة عشرية تقع فى المدى من h0 إلى hFFFF مع وجود &h أو &H تسبقها.	&h10 &HFF
Integer (Octal)	قيمة ثمانية تقع فى المدى من &00 إلى &0177777 مع وجود &0 أو &0 تسبقها.	&010 &0244
Long Integer (Decimal)	مثل الصحيح العشرى (أول نوع فى الجدول) إلا أن المدى يكون فى هذه الحالة من -2,147,483,648 إلى +2,147,483,647.	&250,000 1,255,768
Long Integer (Hexadecimal)	مثل الثابت السادس عشرى (النوع الثانى فى الجدول) إلا أن المدى يكون فى هذه الحالة من &H0 إلى &FFFFFFF. لاحظ ضرورة وجود &.	&H0 &H0DAF
Long Integer (Octal)	مثل الثابت الثمانى (الحالة الثالثة فى الجدول) إلا أن المدى يكون فى هذه الحالة من &00 إلى &03777777777. لاحظ ضرورة وجود &.	&O437 &O4447666
Fixed point	اعداد حقيقية سالبة أو موجبة لها علامات عشرية.	-22.7, 12.44
Floating point	أعداد حقيقية سالبة أو موجبة موجودة فى الصورة الأسية. ويتراوح مداها من -3.37E+38 إلى 3.37E+38.	1.444E+3 256E2 -128.32E-3
Floating point (Double precision)	مثل سابقتها مع استخدام D بدلاً من E فى تحديد الأس. ويتراوح المدى من -1.67D+308 إلى 1.67D+308.	1.444D+30
String	رمز واحد أو أكثر موضوع بين علامتى تنصيص مزدوجة. يمكن أن يشمل كل الرموز الموجودة فى ASCII باستثناء رمز علامة التنصيص المزدوجة ورموز إعادة العربية وتغذية السطر. وأقصى عدد للرموز مسموح به فى السلسلة هو 32,767 رمزاً.	"MOVIE" "Costs \$.00!"

كل هذه الأنواع سألقة الذكر باستثناء آخر نوع وهو نوع السلسلة هي ثوابت عددية. أما نوع السلسلة فيمثل ثوابت غير عددية أو ثوابت حرفية.

التطبيقات

تستخدم الثوابت لحفظ قيم تستخدم أكثر من مرة واحدة في البرنامج ولاتتغير هذه القيم أثناء تنفيذ البرنامج. واختيار الثوابت الحرفية بدلاً من الثوابت العددية هو أكثر من اختياري فتستخدم الثوابت العددية عندما لا يتوقع لها أن تتغير أثناء تنفيذ البرنامج مع استخدام نفس القيمة أكثر من مرة واحدة في نفس البرنامج. لاحظ الأمثلة المقدمة في جزء الوصف من هذا الدرس مثال ذلك الثوابت الرمزية π و FICA والتي يمكن أن تستخدم في أماكن عديدة من البرنامج بدون إعادة كتابة القيمة الموجودة في المتغير. وتكون الثوابت الحرفية خياراً واضحاً عندما يراد استخدام القيمة المحددة مرة واحدة فقط. وأمثلة الثوابت الحرفية الموجودة في جزء الوصف من هذا الدرس مثالية للحسابات مرة واحدة فقط ولعبارات PRINT كذلك.

عملية تقليدية

في برنامج BOX. BAS المقدم في الدرس الثالث يوجد مثال لكيفية استخدام الثوابت. ففي هذا البرنامج تم استخدام الثوابت الرمزية π و π و π و π في حفظ احداثيات الصندوق على الشاشة أثناء تنفيذ البرنامج. ويتغير موضع الصندوق في هذه العملية مع تغيير مجموعة المتغيرات.

١ - ابدأ ببيسك السريع عن طريق كتابة QB والضغط على مفتاح الادخال ارجع إلى الدرس الثاني والملحق B إذا ما كنت في حاجة إلى المزيد من المعلومات عن بدء ببسك السريع.

٢ - اضغط على Alt-F لعرض قائمة File.

٣ - اكتب O لتحميل البرنامج. يظهر صندوق حوار فتح الملف. للمزيد من المعلومات عن صندوق الفتح ارجع إلى الملحق B.

٤ - اضغط على Tab للانتقال إلى الدليل.

٥ - باستخدام مفاتيح الأسهم انقل نقطة البداية إلى البرنامج BOX.BAS واضغط على مفتاح الادخال لاختياره.

٦ - غير قيمة Lrx من 40 إلى 50 وغير قيمة Lry من 10 إلى 14.

٧ - اضغط على Shift-F5 لتنفيذ البرنامج. تستخدم الثوابت الرمزية : Horiz و Vert و Tlc و Trc و Brc و Blc فى رسم الرموز للصندوق. فإذا ما غيرتها إلى قيم أخرى فيتم رسم الصندوق برمز أخرى. لتجربة هذه القيم استخدم العملية الموجودة فى الخطوتين 6 و 7.

٨ - اضغط على أى مفتاح للعودة إلى قائمة البرنامج.

٩ - اضغط على Alt-F واكتب X للخروج من بيسك السريع. اضغط على Tab وعلى قضيب المسافات لاختيار عدم حفظ التغييرات. تجد نفسك قد عدت الآن إلى ملقن DOS.

١٠ - انتقل إلى الدرس المائة والخامس والخمسين للاستمرار فى تسلسل التعلم.

الدرس الرابع والعشرون

دالة COS

الوصف

تعيد دالة COS جيب تمام الزاوية المعرفة بالتعبير العددي، وتكوينها هو كما يلي :

COS (تعبير عددي)

وتكون القيمة التي تعود من الدالة بالتقدير الدائري، فإذا كان التعبير العددي مزوج الدقة فتكون النتيجة مزوجة الدقة كذلك وإلا فإنها تكون فردية الدقة، ويمكن أن يكون التعبير العددي من أي نوع من الأنواع المسموح بها في بيسك السريع.

التطبيقات

تستخدم هذه الدالة عندما يكون مطلوب حساب جيب تمام الزاوية مثل الحسابات المثلثية للرسومات والألعاب، وفيما يلي أمثلة لدالة COS :

```
T = COS(A) * P
PRINT COS(SIN(Pc))
```

عملية تقليدية

توضح العملية التالية استخدام دالة COS، ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Tools
Untitled
This program calculates points equally spaced on the
circumference of a circle.
CLS
PRINT "Calculates points on the circumference of a circle"
INPUT "Enter x,y coordinates of the center of the circle ": x, y
INPUT "Enter the angle of the first point ": Angle
INPUT "Enter number of points to calculate ": Pt
INPUT "Enter the radius of the circle ": radius

Angle = (Angle * 3.14159) / 180
twoPi = 6.28319 / Pt

PRINT "The coordinates : "
FOR cnt = 0 TO Pt - 1
    X1 = x + (radius * COS(Angle + twoPi * cnt))
    Y1 = y + (radius * SIN(Angle + twoPi * cnt))
    PRINT "Point: "; cnt + 1; X1, Y1
NEXT cnt

```

Immediate

Main: Untitled Context: Program not running 00004:019

٢ - نفذ البرنامج. اكتب 10 و 10 كمركز و 45 كقيمة للزاوية و 5 كعدد للنقاط و 2 كنصف قطر.
لاحظ استخدام دالة COS في البرنامج. اضغط على أى مفتاح للعودة إلى البرنامج.

```

Calculates points on the circumference of a circle
Enter x,y coordinates of the center of the circle ? 10,10
Enter the angle of the first point ? 45
Enter number of points to calculate ? 5
Enter the radius of the circle ? 2
The coordinates :
Point: 1 11.41421 11.41421
Point: 2 9.892018 11.78201
Point: 3 8.824624 9.687129
Point: 4 9.687135 8.824623
Point: 5 11.78202 9.892024

Press any key to continue

```

٣ - اختر New من قائمة File واكتب N لإخلاء الشاشة.

٤ - انتقل إلى الدرس الرابع والأربعين للاستمرار في تسلسل التعلم.

الدرس الخامس والعشرون

دالة CSRLIN

الوصف

تعطى هذه الدالة موقع السطر الحالى لنقطة البداية، وتكوينها هو كما يلي :

CSRLIN

ولا توجد أية مؤشرات لهذه الدالة، والقيمة التى تعيدها الدالة هى السطر الذى توجد عنده نقطة البداية فى الوقت الحالى، ويمكن أن تستخدم هذه الدالة فى أحد التعبيرات.

التطبيقات

هذه الدالة هى أحد النوال التى تقدم تحكماً فى نقطة البداية أثناء تنفيذ البرنامج، والنوال الأخرى التى تيسر ذلك هى دالة LOCATE ودالة POS، وفيما يلي بعض الأمثلة :

```
LOCATE CSRLIN + 1, 1
SRow = CSRLIN: SCol = POS
LOCATE 24, 1: PRINT "Please choose from above.": BEEP
LOCATE SRow, SCol
```

تستخدم الدالة فى المثال الأول فى تقديم نقطة البداية إلى السطر التالى، ويستخدم المثال الثانى هذه الدالة مع دالة POS فى التذكرة بموقع نقطة البداية مع طباعة رسالة فى أحد المواقع على الشاشة ثم العودة إلى الموقع الأسمى لنقطة البداية.

عملية تقليدية

لقد أعد هذا البرنامج المقدم فى هذه العملية فى الدرس السادس والثمانين، ويوضح البرنامج استخدام برامج فرعية سبق اختبارها اختباراً جيداً لأغراض خاصة، ويقبل البرنامج مدخلات من لوحة المفاتيح طبقاً لمؤشرات محددة، وتحدد المؤشرات السطر والعمود اللذين يجب أن تقبل المدخلات عندهما سلسلة الملقن ومجموعة من الرموز التى تمثل مدخلات صحيحة، وتقوم أنت فى هذه العملية بتعديل البرنامج لطباعة رسالة خطأ فى أحد المواقع على الشاشة والعودة إلى الملقن لادخال مدخلات، ابدأ بتحميل بيسك السريع.

١ - اختر Open وحمل البرنامج الذي سبق اعداده في الدرس السادس والثمانين

.LRTRIM.BAS

٢ - عدل الملاحظات ودالة GCH\$ إلى ما هو مبين في السرد التالي :

```
File Edit View Search Run Debug Tools File Help
LRTRIM.BAS
'This program demonstrates the use of the CSRLIN function.
'The program was created in Module 89; here the function GetChar
'is modified to print a message when the input is incorrect.

DECLARE FUNCTION Gch$(UCH AS STRING, X AS INTEGER, Y AS INTEGER, P AS STRING)
DIM UCH AS STRING * 30, P AS STRING * 75
CLS
LOCATE 23, 1
PRINT "E)dit / C)reate / D)elete / Q)uit ?"

ValidCh = "EDCQ"
Prompt = "Enter selection"
Selection$ = Gch$("EDCQ", 24, 1, Prompt)
```

```
File Edit View Search Run Debug Tools File Help
LRTRIM.BAS:Gch
FUNCTION Gch$(UCH AS STRING, X AS INTEGER, Y AS INTEGER, P AS STRING)
DIM Choice AS STRING
Prompt = LTRIM$(RTRIM$(P))
Choice = ""
LOCATE X, Y
PRINT P;

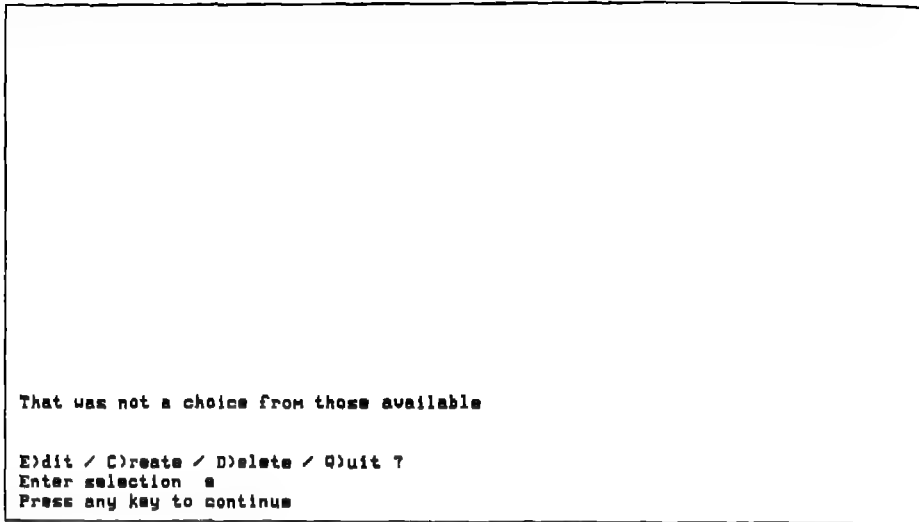
DO
Choice = INKEY$
IF (INSTR(ValidCh, UCASE$(Choice)) = 0) THEN
Crow = CSRLIN: CCol = POS(0)
LOCATE 20, 1: Choice = ""
PRINT "That was not a choice from those available": BEEP
LOCATE Crow, CCol
END IF
LOOP UNTIL (Choice <> "")

LOCATE X, (Y + 2 + LEN(P))
PRINT Choice;
Gch$ = Choice
END FUNCTION

----- Immediate -----
Main: LRTRIM.BAS  (Type*) Program not running  (000) (0) (000)
```

٣ - نفذ البرنامج. اكتب X لتظهر الرسالة. اكتب Q لإنهاء البرنامج لاحظ استخدام هذه الدالة

في حفظ موقع السطر الحالي. وفيما يلي عينة للمخرجات :



- ٤ - ارجع إلى البرنامج واحفظه على أنه برنامج نصي تحت اسم CSRLIN.BAS
- ٥ - انتقل إلى الدرس المائة وسبعة للاستمرار في تسلسل التعلم.

الدرس السادس والعشرون

دوال CVD و CVI و CVL و CVS

الوصف

دوال CVD و CVI و CVL و CVS هي عكس الدوال MKD\$ و MKI\$ و MKL\$ و MKS\$. وتحول هذه الدوال قيم السلاسل الناتجة باستخدام الدوال MKD\$ و MKI\$ و MKL\$ و MKS\$ إلى قيم عددية. وتكوينها هو كما يلي :

```
CVD(8 byte string)
CVI(2 byte string)
CVL(4 byte string)
CVS(4 byte string)
```

وتحول دالة CVD سلسلة مكونة من 8 بايت سبق انتاجها بواسطة دالة MKD\$ إلى قيمة عددية مزوجة الدقة.

وتحول دالة CVI سلسلة مكونة من 2 بايت سبق انتاجها بواسطة دالة MKI\$ إلى قيمة عددية صحيحة.

وتحول دالة CVL سلسلة مكونة من 4 بايت سبق انتاجها بواسطة دالة MKL\$ إلى قيمة عددية صحيحة طويلة.

وتحول دالة CVS سلسلة مكونة من 4 بايت سبق انتاجها بواسطة دالة MKS\$ إلى قيمة عددية فردية الدقة.

تستخدم هذه الدوال مع عبارة FIELD في تحميل بيانات داخل المتغيرات. وتستخدم هذه الدوال في إبطال عمل التحويلات التي تعد بواسطة الدوال MKD\$ و MKI\$ و MKL\$ و MKS\$. وفيما يلي مثال لاستخدامها :

```
OPEN "SalesTx.Dat" FOR RANDOM AS #3 LEN = 42
...
FIELD #3 20 AS ItemName$, 10 AS Qty$, 12 AS SalesTx$
...
GET #3
Quantity = CVI(Qty$): SalesTx! = CVS(SalesTx$)
PRINT ItemName$, Quantity, SalesTx!
```

يوضح المثال السابق كيفية استخدام دالتي CVI و CVS في تحويل السلاسل إلى قيمها العددية. ويمكن تحقيق نفس التأثير عن طريق انتاج أنواع تكوينات وملفات يقوم المستخدم بتعريفها لهذه الأنواع. ويوضح المثال التالي كيفية استخدام عبارات TYPE و END TYPE في تحقيق نفس الشيء. لاحظ سهولة وبساطة قراءة الشفرة باستخدام هذه الطريقة.

```
TYPE SalesTx
  ItemName AS STRING * 20
  Qty      AS STRING * 10
  STax     AS STRING * 12
END TYPE
DIM STRec AS SalesTx
OPEN "SalesTx.Dat" FOR RANDOM AS #3 LEN = LEN(STRec)
PRINT #3,STRec
```

عملية تقليدية

توضح هذه العملية الدوال CVD و CVI و CVS. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :

```
File Edit View Search Run Debug Calls F1=Help
Untitled
This program illustrates the use of CVS, CUD, and CVI functions.
The program reads in a file created in Module 88, CUSTZ.FIL.

OPEN "CUSTZ.FIL" FOR RANDOM AS #2
FIELD #2, 25 AS CName$, 2 AS CNum$, 8 AS CreditLim$, 4 AS LastInv$
RecCnt = 2
GET #2, 1

DO WHILE NOT EOF(2)
  PRINT CName$; CVI(CNum$); CUD(CreditLim$); CUS(LastInv$)
  GET #2
LOOP

CLOSE #2
```

Immediate

Menu: (Untitled) Context: Program not running 00010:040

٢ - نفذ البرنامج. لاحظ استخدام النوال سالفة الذكر في البرنامج.

Mission Impossible Inc.	1233	300000	12000
Last Resort Motel	888	25000	3000
Income Only Corp.	3433	1000000	120000

Press any key to continue

٣ - ارجع إلى البرنامج واختر New ولا تحفظ هذا البرنامج.

٤ - انتقل إلى الدرس التاسع والثمانين للاستمرار في تسلسل التعلم.

الدرس السابع والعشرون

عبارة DATA

الوصف

تستخدم عبارة DATA في دعم عبارة READ بعناصر البيانات، وتكوينها كما يلي :

DATA (قائمة ثوابت)

الوصف	الجزء
كلمة من كلمات بيسك السريع المحجوزة، قائمة بقيم الثوابت مفصولة عن بعضها البعض بواسطة فواصل، ويمكن أن تكون السلاسل عديدة أو تكون ثوابت حرفية.	DATA قائمة ثوابت

إذا ظهرت ثوابت رمزية (ثوابت معرفة في عبارة CONST) في عبارة DATA فتفسر هذه الثوابت كقيمة سلسلة وليست كالقيمة المعرفة في عبارة CONST وفيما يلي مثال لذلك :

```
CONST MaxList = 200
READ Val1,Val2
DATA 100,MaxList
```

إذا نفذت عبارة READ فيفسر عنصر البيانات الثاني (MaxList) كثابت سلسلة (ثابت حرفي) وليس كالقيمة العددية 200.

اعتبارات أخرى : يمكن أن تحتوى عبارة DATA على أى عدد من الثوابت يمكن كتابته في سطر واحد، ويمكن أن تظهر العديد من عبارات DATA في أسطر متتالية، يمكن أن تحتوى عبارة DATA على أكثر من قيمة واحدة تتطلبها عبارة READ، وتستمر عبارة READ من آخر عنصر بيانات لم يسبق قراءته من القائمة، ويجب أن تحتوى عبارة DATA على عناصر بيانات لا يقل عددها عن عدد المتغيرات الموجودة في عبارة READ، وإذا كانت هناك عناصر بيانات أقل من المتغيرات فيحدث خطأ وقت التشغيل.

التطبيقات

تستخدم عبارة DATA مع عبارة READ فقط. وعلى هذا فيجب أن تظهر مرة واحدة على الأقل في أى برنامج توجد به إحدى عبارات READ. ويجب أن تتفق كذلك مع متطلبات التكوين واللغة والمنطق. وفيما يلي بعض الأمثلة :

مثال ١

```
READ Qty%,Rate#,Invoice$
DATA 2300,190.23,QR9989
```

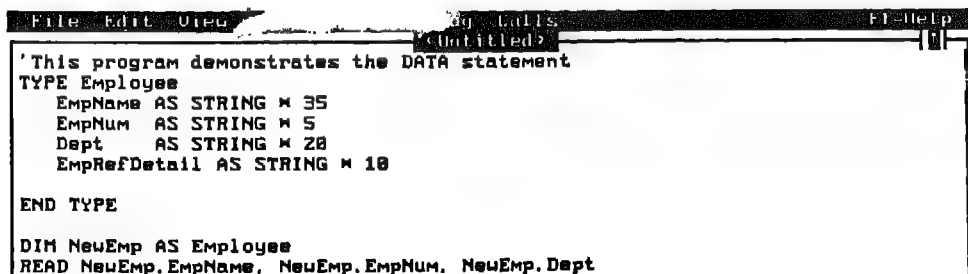
مثال ٢

```
TYPE Employee
  EmpName AS STRING*35
  EmpNum AS STRING*5
  Dept AS STRING*20
  EmpRefDetail AS STRING*10
END TYPE
DIM NewEmployee AS Employee
READ NewEmployee.EmpName,NewEmployee.EmpNum,NewEmployee.Dept
DATA Jon X. Hancock,WS3332,Food&Bev
```

عملية تقليدية

يوضح البرنامج الموجود فى هذه العملية استخدام عبارة DATA. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :



```
'This program demonstrates the DATA statement
TYPE Employee
  EmpName AS STRING * 35
  EmpNum AS STRING * 5
  Dept AS STRING * 20
  EmpRefDetail AS STRING * 10
END TYPE
DIM NeuEmp AS Employee
READ NeuEmp.EmpName, NeuEmp.EmpNum, NeuEmp.Dept
```



```
CLS
PRINT "This is a demonstration of the DATA statement"
PRINT : PRINT
PRINT "Employee Name      : "; NewEmp.EmpName
PRINT "Employee Number    : "; NewEmp.EmpNum
PRINT "Department         : "; NewEmp.Dept
PRINT "Employee Reference : "; NewEmp.EmpRefDetail
DATA Jon X. Hancock, US332, Food&Bev
```

----- Immediate -----

Name: (Untitled) Context: Program not running 00001:037

٢ - نفذ البرنامج ولاحظ استخدام عبارة DATA في البرنامج.

```
This is a demonstration of the DATA statement

Employee Name      : Jon X. Hancock
Employee Number    : US332
Department         : Food&Bev
Employee Reference :

Press any key to continue
```

٣ - اضغط على أى مفتاح للعودة إلى البرنامج. اضغط على Alt-F ثم اضغط على مفتاح الانخال واكتب N لإخلاء الشاشة.

٤ - انتقل إلى الدرس المائة والتاسع والأربعين للاستمرار في تسلسل التعلم.

الدرس الثامن والعشرون

دالة وعبارة DATE\$

الوصف

يمكن استخدام هذه الكلمة الرئيسية كعبارة أو كدالة وذلك مثل الكلمة الرئيسية \$TIME. وكعبارة تضع هذه الكلمة تاريخ الكمبيوتر أما كدالة فإنها تعيد التاريخ الحالى للكمبيوتر. وتكوينها فى كل من الحالتين هو كما يلى :

تكوين العبارة:

DATE\$ = تعبير سلسلة

الوصف	الجزء
كلمة من كلمات بيسك السريع المحجوزة. يمكن لتعبير السلسلة أن يكون متغيراً أو تعبيراً ثابتاً. ويجب أن يكون فى أحد الصور التالية : mm-dd-yy أو mm-dd-yyyy أو mm/dd/yy أو yy/mm/dd/yyyy حيث mm هى الشهر و dd هى اليوم و yy أو yyyy هى السنة.	DATE\$ تعبير سلسلة

تكوين الدالة : وتكوين الكلمة كدالة هو كما يلى :

DATE\$

وتعيد هذه الدالة سلسلة فى الصورة mm-dd-yyyy تعبر عن التاريخ الحالى للكمبيوتر.

التطبيقات

تستخدم هذه الكلمة أساساً فى قراءة ووضع تاريخ نظام الكمبيوتر. وهى مفيدة جداً عند كتابة تقارير باستخدام الطابع. كما يمكن أن تستخدم كذلك فى حفظ تتبع تجديدات الملفات عن طريق إضافة سجل آخر إلى الملف مع وجود تاريخ التجديد عليه. كما توجد تطبيقات عديدة أخرى كذلك. وفيما يلى بعض الأمثلة :

: أمثلة لعبارة DATE\$

```
DATE$ = "12/31/88"
DATE$ = "01/22/1987"
DATE$ = "02-02-80"
DATE$ = "11-01-79"
```

: أمثلة لدالة DATE\$

```
CurrDate$ = DATE$
PRINT DATE$
```

سبابة تقليدية

توضح هذه العملية استخدام الكلمة كدالة. ابدأ بتحميل بيك السريع.

١ - اكتب البرنامج التالي :

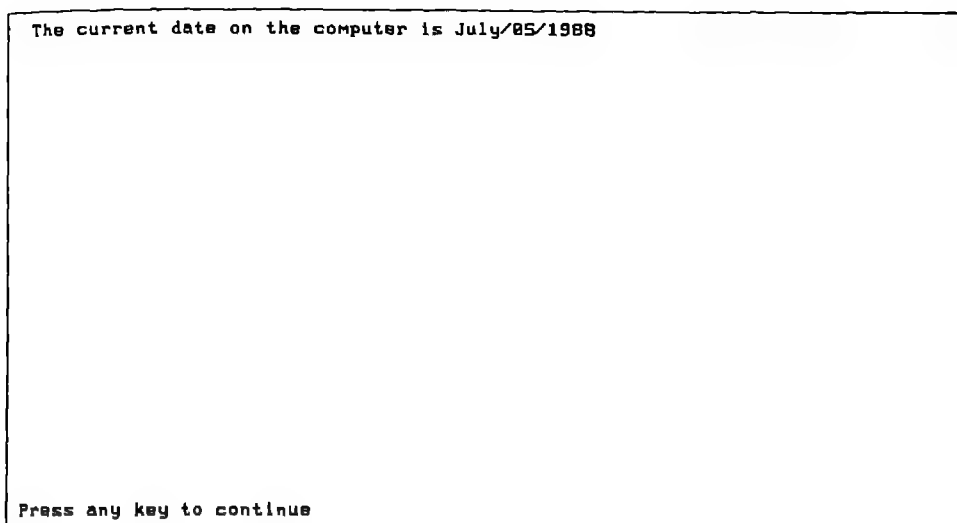
```

File Edit View Search Run Debug Calls...
DATE.DAS
' This program demonstrates the DATE$ verb
DIM Months$(12)
FOR m = 1 TO 12: READ Months(m): NEXT m
CLS
DS = DATE$
PRINT "The current date on the computer is "; Months(VAL(DS)); "/";
PRINT MID$(DS, 4, 2); "/"; MID$(DS, 7, 4)

DATA January, February, March, April, May, June, July
DATA August, September, October, November, December

Immediate
Main- Date.Das  Context- Program not running  000001-0001
```

٢ - نفذ البرنامج ولاحظ المخرجات واستخدام دالة DATE\$ في البرنامج.



٣ - اضغط على أى مفتاح للعودة إلى البرنامج. اضغط على Alt-F واضغط على مفتاح الإدخال واكتب N لإخلاء الشاشة.

٤ - انتقل إلى الدرس الثامن والأربعين للاستمرار فى تسلسل التعلم.

الدرس التاسع وعشرون

عبارة DECLARE

الوصف

تتسبب هذه العبارة في أن يشير البرنامج الرئيسى إلى برامج بيسك سريع فرعية وإلى إجراءات فرعية ويبدأ عملية التأكد من نوع المؤشر. وتستخدم هذه العبارة للإشارة إلى إجراءات في بيسك السريع وكذلك إجراءات مكتوبة بكل من لغتي المجمع و C. وفيما يلي تكوين العبارة. (يحدد FUNCTION | SUB استخدام احدى الكلمتين المحجوزتين وليس الاثنتين معاً).

إجراءات بيسك السريع:

```
DECLARE FUNCTION|SUB name (parameterlist)
```

غير إجراءات بيسك السريع:

```
DECLARE FUNCTION|SUB name CDECL ALIAS "Alias name"(parameterlist)
```

إجراءات بيسك السريع : جزء name هو اسم للدالة FUNCTION أو للبرنامج الفرعى الذى يستخدم فى عبارات التحديد واستدعاءات الإجراء، ويكون الاسم معرفاً صحيحاً من معرفات بيسك السريع ويمكن أن يشغل حتى 40 خانة كحد أقصى. وتأخذ قائمة المؤشر param-eter list التكوين التالى :

```
Var1 AS type, var2 AS type,...
```

جزئى var1 و var2 عبارة عن متغيرات من متغيرات بيسك السريع ويعرف جزء AS type نوع البيانات للمتغيرات. ويمكن أن يكون هذا النوع من النوع البسيط أو النوع الذى يعرفه المستفيد. وغير مسموح باستخدام سلاسل ثابتة الطول. وعندما يكون المتغير منظومة فيلى المتغير قوسان فارغان على النحو التالى :

```
var1(), var2(),...
```

ويحدد شكل العبارة ما إذا كان نوع القوائم قد اختبر أم لا، ويوضح الجدول التالى ذلك :

التكوين	كيفية أداء اختبار نوع القائمة
<p>DECLARE SUB name1</p> <p>DECLARE SUB name ()</p> <p>DECLARE SUB name1</p> <p>(var1 AS INTEGER)</p>	<p>لا يسرد أى مؤشرات أساسية. ولا يستخدم أى اختبار للنوع.</p> <p>لا يوجد فيه أى مؤشرات. محاولة المرور عند المؤشر لهذا البرنامج الفرعى تتسبب فى حدوث خطأ.</p> <p>لا يوجد مؤشر فى SUB. ويحدث اختبار لنوع القائمة.</p>

وتستخدم عبارة DECLARE عند استدعاء برنامج فرعى بدون الكلمة المحجوزة CALL أو عند استدعاء برنامج فرعى فى جزء آخر. وينتج ببسك السريع هذه العبارات للجزء قبل أن يكتب الجزء فى الملف. ومثل هذه العبارات التى تظهر فى أحد الأجزاء تكون مخصصة لهذا الجزء فقط وتؤثر على محتوياته.

غير اجراءات ببسك السريع : فى هذا التكوين جزء name هو اسم صحيح من اسماء متغيرات ببسك السريع ويمكن أن يشغل حتى 40 خانة كحد أقصى. ويحدد جزء CDECL أن البرنامج الفرعى المستدعى، أو الدالة المستدعاة، هو برنامج فرعى مكتوب بلغة C. واستخدام ذلك يؤثر على كيفية البحث فى المكتبات وفى ملفات التشغيل object files للبرنامج الفرعى. وعند استخدام CDECL يتحول الاسم إلى حروف صغيرة ويدمج رمز الشرطة التى تقع تحت الحرف قبله كما تمرر القائمة كذلك من اليمين إلى اليسار بدلاً من معدل ببسك السريع المعتاد من اليسار إلى اليمين. ويحدد جزء ALIAS أن البرنامج الفرعى له اسم مختلف فى المكتبة أو فى ملفات التشغيل. وجزء "Alias name" هو اسم البرنامج الفرعى الموجود فى المكتبة. وعند استخدام الكلمة المحجوزة ALIAS يستخدم "Alias name" فى البحث فى المكتبة وفى ملفات التشغيل. ولقائمة المؤشرات الشكل التالى :

BYVAL|SEG var1 AS type, BYVAL|SEG var2 AS type...

يحدد جزء BYVAL|SEG كيفية تمرير القائمة إلى البرنامج الفرعى. وتستخدم الكلمة المحجوزة BYVAL إذا ما تم تمرير القائمة طبقاً للقيمة. وتستخدم الكلمة المحجوزة SEG إذا ما تم تمرير القائمة كعنوان مجزأ. وأجزاء var1 و var2 و AS type هى نفسها مثلما سبق وصفه فى القسم الخاص بتوضيح اجراءات ببسك السريع. ويتسبب استخدام الكلمة المحجوزة

ANY بدلاً من AS type فى اجتياز اختبار نوع القائمة. وعند استخدام BYVAL فى قائمة المؤشرات يمكنك ألا تستخدم الكلمة المحجوزة ANY.

يعتمد اختبار النوع على شكل عبارة DECLARE. ارجع إلى القسم الخاص بهذه العبارة فى اجراءات بيسك السريع.

تخزين المتغير

يناقش هذا القسم كيفية تخزين بيسك السريع للمتغيرات. وتفيد معرفة ذلك عند استخدام برمجة بلغات مختلطة أو عند استخدام احدى النوال التالية :

SADD
SETMEM
VARPTR
VARSEG
VARPTR\$

يخزن بيسك السريع المتغيرات فى منطقة تسمى DGROUP (جزء البيانات التقليدى للبرنامج) أو كعناوين بعيدة. وعند تنفيذ البرنامج كبرنامج قائم بذاته تخزن كل المتغيرات البسيطة والمنظومات الاستاتيكية \$STATIC والمنظومات الديناميكية \$DYNAMIC لسلاسل المتغيرات فى DGROUP. ويمكن الاشارة إلى ذلك باستخدام عناوين ومشيريات قريبة. أما الأنواع الأخرى للمنظومات الديناميكية فتخزن كعناوين بعيدة. وعند تنفيذ البرنامج فى بيئة بيسك السريع توضح كل المتغيرات البسيطة والمنظومات الاستاتيكية فى عبارة مشاركة وتخزن منظومات سلاسل المتغيرات فى DGROUP وتخزن بقية المنظومات الأخرى كلها كأشياء objects بعيدة.

والأشياء التى يمكن أن تقود إلى متغيرات تنقل فى الذاكرة هى ما يلى :

- اشارة إلى ثابت سلسلة أو تعبير سلسلة.

- استدعاء DEF FN أو FUNCTION.

- استخدام نوال سلاسل أو نوال مرتبطة بالذاكرة.

- اشارة إلى منظومة محدد لها أبعاد ضمناً.

التطبيقات

هذه العبارة تمثل آلية تحكم لضمان أن كل البرامج الفرعية التي يشار إليها في برنامج بيسك السريع متاحة وأن المؤشرات التي تمرر تكون متوافقة مع المؤشرات الأساسية. وينتج بيسك السريع هذه العبارات (عبارات DECLARE) في الحالات التي تكون فيها البرامج الفرعية جزءاً من المقطع، وعندما لا تكون كذلك فيجب أن تكتب عبارة توضيح DECLARE في ملف \$INCLUDE واحد. ومع هذه الطريقة لا تقلق من وجود كل عبارات التوضيح هناك. وعبارات التوضيح هذه ضرورية عند استدعاء البرنامج الفرعي بدون استخدام الكلمة المحجوزة CALL. ويقوم بيسك السريع بإدخال عبارات التوضيح حتى إذا كان المقطع يستخدم كلمة CALL المحجوزة في عبارة الاستدعاء. وفيما يلي بعض الأمثلة لهذه العبارة :

```
DECLARE FUNCTION MoveRight$(AnyStr$)
DECLARE FUNCTION MoveLeft$
DECLARE FUNCTION CenterText$ ( )
```

تشمل عبارة التوضيح الأولى قائمة مؤشرات أساسية وينفذ اختبار النوع. ولا توجد قائمة مؤشرات أساسية في عبارة التوضيح الثانية ولا ينفذ اختبار النوع. ولا يعنى هذا أن FUNC-TION Moveleft\$ ليست لها أية مؤشرات، وإنما لا ينفذ اختبار النوع على القوائم التي تمر إلى الدالة فقط. وتشمل عبارة التوضيح الثالثة قائمة مؤشرات فارغة تتسبب في ظهور رسالة خطأ إذا ما حدثت محاولة لتمرير أى مؤشرات.

عملية تقليدية

توضح هذه العملية كيفية استخدام عبارة التوضيح في أحد البرامج. وقد سبق إعداد البرنامج في الدرس الحادى والسبعين. ابدأ بتحميل بيسك السريع.

١ - اختر Open وحمل البرنامج LUBOUND.BAS.

٢ - لاحظ عبارة التوضيح في البرنامج. عدل عبارات CALL إلى ماهى عليه في السرد التالى :


```

File Edit View Search Run Debug Calls F1=Help
LBOUND.BAS

' The following program demonstrates the use of the UBOUND and LBOUND
' statements. The program loads two sets of array values and
' finds the minimum and maximum values in those arrays.

DECLARE SUB FindMinMax (A%( ), MinVal, MaxVal)

Max = 15
DIM A%(Max)
GOTO Start

LoadArray:
  FOR Cnt = 1 TO Max
    READ A%(Cnt)
  NEXT
  RETURN

Start:
  CLS
  GOSUB LoadArray
  PRINT "First pass"
  FindMinMax A%( ), MinVal, MaxVal
  READ Max
  REDIM A%(Max)
  GOSUB LoadArray
  PRINT "Second pass"
  FindMinMax A%( ), MinVal, MaxVal

DATA 12,23,33,43,1.56,98,656,323,44,9.88,67,54,10
DATA 10
DATA 8.89,76,54,23,32,12,4,33,54
SUB FindMinMax (A%( ), MinVal, MaxVal)
  MinVal = A%(1): MaxVal = A%(1)
  FOR Cnt = LBOUND(A%) + 1 TO UBOUND(A%)
    IF MinVal > A%(Cnt) THEN
      MinVal = A%(Cnt)
    END IF
    IF MaxVal < A%(Cnt) THEN
      MaxVal = A%(Cnt)
    END IF
  NEXT
  PRINT "Minimum value in array: "; MinVal, "Maximum value in array: "; MaxVal
END SUB

Immediate

Main: LBOUND.BAS Context: Program not running 00001.001

```

٢ - نفذ البرنامج. تظهر شاشة المخرجات على النحو التالي :

```
First pass
Minimum value in array: 1 Maximum value in array: 656
Second pass
Minimum value in array: 4 Maximum value in array: 89
```

Press any key to continue

- ٤ - ارجع إلى البرنامج مع اخلاء الشاشة دون أن تحفظ البرنامج.
- ٥ - انتقل إلى الدرس المائة والرابع والأربعين للاستمرار في تسلسل التعلم.

الدرس الثلاثون

عبارة DEF FN

الوصف

تقوم عبارة DEF FN بتعريف احدى النوال، وتوجد طريقتان لعمل ذلك وتكوين العبارة هو كمايلي :

التكوين الأول (سطر واحد) :

```
DEF FNname (parameter list) = expression
```

التكوين الثاني (أسطر متعددة) :

```
DEF FNname (parameter list)
    FNname = expression
END DEF
```

جزء name المصاحب للكلمة المحجوزة FN يعطى للدالة اسمها، و name هو اسم متغير صحيح فى بيسك السريع (يمكن أن يصل طوله حتى 40 خانة كحد أقصى محدداً أى نوع لرمز التوضيح)، وفيما يلي مثال لذلك :

```
DEF FNTOKEN$(InString$) = MID$(InString$,1,10)
```

جزء parameter list هو قائمة بأسماء متغيرات مع استخدام الفواصل بينها، ويمكن أن يوجد فى قائمة الأسماء توضيح اختياري لنوع المتغيرات كجزء من القائمة، وفيما يلي مثال لذلك :

```
(TempInC AS SINGLE, TempInF AS SINGLE, Count AS INTEGER)
```

وتحدد القيم لهذه المتغيرات من البيئة المنادية، وتمرر كل المؤشرات طبقاً للقيمة، وهناك طريقتان لتمرير المؤشرات إلى الإجراء أو إلى الدالة وذلك طبقاً للقيمة أو طبقاً للدليل، وكل من الطريقتين مشروحة فى المقاطع التالية.

المرور طبقاً للقيمة : فى هذه الحالة تنسخ محتويات المتغيرات من البيئة المنادية داخل قائمة المؤشرات فى الدالة، وتكون هذه المتغيرات محلية للدالة نفسها ولا تؤثر على ذلك على أى متغير من المتغيرات على المستوى الشامل.

المورد طبقاً للدليل : فى هذه الحالة تتأثر المتغيرات فى البيئة المنادية بأى تعديل يجرى على المتغيرات فى قائمة المؤشرات للدالة.

ويحدث فى التكوينين 1 و 2 حساب قيمة التعبير expression وتحديد النتيجة لاسم الدالة FN name. ويشمل جسم الدالة تعبيراً فى التكوين 1. أما فى التكوين 2 فيكون التعبير جزءاً فقط من التحديد. وهذه هى طريقة اعادة الدالة للنتيجة إلى البيئة المنادية. وعند ترك ذلك، تعيد الدالة صفراً لعبارة DEF FN المتعدية وفراغاً لعبارة DEF FN الحرفية (السلسلة).

اعتبارات أخرى : فيما يلى قائمة بالأشياء التى يجب تذكرها عن عبارة DEF FN :

- يجب أن تعرف الدالة قبل أن يمكن استخدامها. فإذا ما لم يتحقق ذلك فتظهر رسالة تفيد بأن الدالة غير معرفة.

- لا يمكن أن تتداخل النوال مع بعضها البعض. فلا يمكن أن تظهر عبارة DEF FN داخل عبارة DEF FN أخرى.

- لا يمكن لدالة معرفة بواسطة عبارة هذا الفصل أن تكون لها اعادة ذاتية، فلا تستطيع الدالة استدعاء نفسها.

- تستخدم عبارة EXIT DEF للخروج بصفة دائمة من DEF FN.

- يجب أن توضح عبارة DEF FN المشار إليها بدليل فى نفس المقطع. فهى محلية بالنسبة إلى المقطع (ملف المصدر).

- كن حذراً عند اعادة اصدار أوامر تعبيرات داخل بيئة وقت تشغيل البيسك لأغراض الكفاءة. يمكن لبيسك السريع أن يعيد اصدار أمر تعبير دون أن يخطر المبرمج بذلك. ويمكن تجنب الآثار الجانبية لهذا عن طريق عزل استدعاء الدالة. وفيما يلى مثال لذلك :

```
DEF FNMult
  InNum = 3
  FnMult = InNum * 10
END DEF
InNum = 1
PRINT FNMult + 23 + InNum
```

وعندما يعاد أمر اصدار تعبير فى عبارة PRINT مثل استدعاء FNMult بعد 23 + InNum تكون النتيجة 54 بدلاً من 56.

- المتغيرات غير الموضحة على أنها استاتيكية داخل عبارة DEF FN وليست جزءاً من قائمة المؤشرات تكون عاملة في هذا المقطع.

- القيم التي تعود بواسطة عبارة DEF FN تكون من نفس نوع الدالة بغض النظر عن تعبير المصدر. عندما تتحدد قيمة عددية لدالة سلسلة أو العكس عندما تتحدد قيمة سلسلة لدالة عددية تظهر رسالة بعدم مواسمة الدالة.

التطبيقات

تزداد فائدة عبارة DEF FN في عملية عزل أجزاء من شفرة سبق اختبارها اختباراً جيداً وتنفيذ أنشطة معرفة تعريفاً جيداً بحيث إن البرنامج يكون مرتباً وسهلاً في قراءته، واعداد مكثبات مصدر بمثل هذه النوال مفيد للغاية، وفيما يلي بعض الأمثلة :

عبارة DEF FN في سطر واحد :

```
DEF FNChis% = ([D - E]^2)/E
DEF FNCirc% = (PiValue * Radius)^2
DEF FNTruncStr$ = RIGHT$(AnyStr$,10)
```

عبارة DEF FN في عدة أسطر :

مثال ١

```
DEF FNStepUp(InValue)
    FnStepUp = InValue * 2
END DEF
```

مثال ٢

```
DEF FNPayrollUpdate(Name$, Address$, NewPScale)
    ..
    FNPayrollUpdate = 0
END DEF
```

يعيد هذا المثال صفراً عندما يتم اجراء العملية بنجاح.

عملية تقليدية

هذه العملية توضح استخدام عبارة DEF FN. ابدأ بتحميل ببسك السريع.

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls File Help
Untitled
'This program demonstrates the use of the DEF FN statement. The program
'generates the period of a pendulum given the length.

CONST pi = 3.14159
GOTO Start
DEF FnPeriod! (length)
    FnPeriod! = (2 * pi) * SQR(length / 981)
END DEF

Start:
CLS
INPUT "Enter the length of the pendulum in centimeters": length
PRINT "The period is "; FnPeriod!(length)

Immediate

Main: Untitled Context: Program not running 0001:000

```

٢ - نفذ البرنامج. لاحظ استخدام عبارة DEF FN في البرنامج.

```

Enter the length of the pendulum in centimeters? 28
The period is .8971395

Press any key to continue

```

٣ - ارجع إلى البرنامج مع اخلاء الشاشة دون أن تحفظ البرنامج.

٤ - انتقل إلى الدرس التاسع والعشرين للاستمرار في تسلسل التعلم.

الدرس الحادى والثلاثون

عبارة DEF SEG

الوصف

ذاكرة الكمبيوتر مقسمة إلى قطاعات كل منها يتكون من 64 كيلو بايت. ويتكون عنوان الذاكرة من جزئين هما القطاع segment والفرع offset. ويقوم الكمبيوتر بإيجاد البيانات الموجودة فى الذاكرة عن طريق الاتصال بالقطاع أولاً ثم البحث عن البيانات داخل القطاع فى أحد أفرع هذا القطاع. وعندما تنفذ البرامج فإنها تحتل قطاعات معينة وتنتج عناوين بيانات تبدأ بقطاعات بدايتها.

وعبارة DEF SEG تضع عنوان القطاع الحالى كقيمة لقطاع جديد. وتكوينها هو كما يلى:

DEF SEG = address

جزء address هو تعبير عددي تقع قيمته فى المدى من 0 إلى 65,535. ويوضع هذا العنوان كقطاع ذاكرة جديد لكل من PEEK و POKE و BLOAD و BSAVE و CALL و ABSOLUTE. وعندما يقع العنوان خارج المدى المحدد تظهر رسالة تفيد بأن استدعاء الدالة غير مسموح به. وعندما يحذف جزء العنوان address فيحدث تحديد لقطاع تقليدي كقطاع بيسك السريع للبيانات.

التطبيقات

تنبيه

هذه العبارة عبارة جديدة ويجب أن تستخدم مع الحذر الشديد. حيث إنه إذا ما تم اختيار قطاع غير مناسب ونفذت سلسلة من PEEK و POKE أو تمت أى محاولة للاتصال المباشر للذاكرة فتكون النتائج وخيمة.

والاستخدام الآمن لهذه العبارة عادة ما يكون مع عبارات BLOAD و BSAVE. وفيما يلى بعض الأمثلة لهذه العبارة :

مثال ١

```
DEF SEG = 0
PEEK(...)
POKE ...
```

مثال ٢

```
DEF SEG = 1024
...
DEF SEG
```

يوضح هذا المثال استخدام العبارة بدون عنوان، وهذا يعيد القطاع إلى قطاع بيسك السريع للبيانات.

عملية تقليدية

توضح العملية عبارة DEF SEG، ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :

```
File Edit View Search Run Debug Calls F1-Help
Untitled
' This program demonstrates the use of the DEF SEG statement.
' The program finds the amount of installed memory on the mother
' board, not extended or expanded, etc.
' The program should work on IBM PCs and full compatibles.

' find status in ram
DEF SEG = &H40
' memory size
totmem = 256 * PEEK(&H14) + PEEK(&H13)
CLS
PRINT "Total installed memory "; totmem; " Kbytes"
DEF SEG

Immediate

Main: 'Untitled' Context: Program not running 00012:000
```

٢ - نفذ البرنامج. لاحظ استخدام عبارة DEF SEG في البرنامج.

Total installed memory 640 Kbytes

Press any key to continue

٣ - ارجع إلى البرنامج مع اخلاء الشاشة نون أن تحفظ البرنامج.

٤ - انتقل إلى الدرس المائة للاستمرار في تسلسل التعلم.

الدرس الثانى والثلاثون

عبارات DEFDBL و DEFINT و DEFLNG و DEFSNG و DEFSTR الوصف

تعرف هذه العبارات مجموعة من الحروف بأنها أنواع بيانات بسيطة، وتكوينها هو كما يلى :

```
DEFDBL letter range
DEFINT letter range
DEFLNG letter range
DEFSNG letter range
DEFSTR letter range
```

جزء letter range هو مدى الحروف الأبجدية التى تعرف بأنها من نوع معين، وتعرف DE- FDBL مدى الحرف بأنه مزوج الدقة، أما عبارة DEFINT فتعرف مدى الحرف بأنه صحيح، وتعريف عبارة DEFLNG أن مدى الحرف صحيح وطويل، أما عبارة DEFSNG فتعرف أن مدى الحرف فردى الدقة، وتعريف عبارة DEFSTR أن مدى الحرف من نوع السلسلة.

ما يعنيه هذا التعريف هو أن أى متغير يبدأ بحرف يقع فى المدى يكون من هذا النوع وذلك بدون عمل توضيح صريح لهذا المتغير. وتأخذ رموز توضيح النوع % و & و ! و # و \$ أسبقية على عبارات DEFtype (وهى DEFDBL و DEFINT و DEFLNG و DEFSNG و DEF-STR). ولا تؤثر هذه العبارات على عناصر السجل، ولا تؤثر حالة الحرف فى شيء، أى إن ما يلى كله متساوى فى تعريفه :

```
DEFINT A-F
DEFINT a-f
DEFINT A-f
DEFINT a-f
```

ويمكن تعريف أكثر من مدى حرف واحد بأنه من نوع معين وذلك باستخدام الفاصلة كفاصل، مثال ذلك ما يلى :

```
DEFSTR a-e, q-s, x-z
```

وبمجرد تعريف مدى حرف بأنه من نوع معين فلا يمكن تغيير ذلك باستخدام أى عبارة أخرى من عبارات DEFtype.

التطبيقات

عبارات DEFtype تقدم وسيلة مريحة لتوضيح النوع بكميات كبيرة. وهى تشبه عبارات توضيح النوع فى لغة الفورتران. كما تؤثر عبارات DEFtype كذلك على دوال DEF FN وتوضيحات FUNCTION. وفيما يلى مثال لذلك :

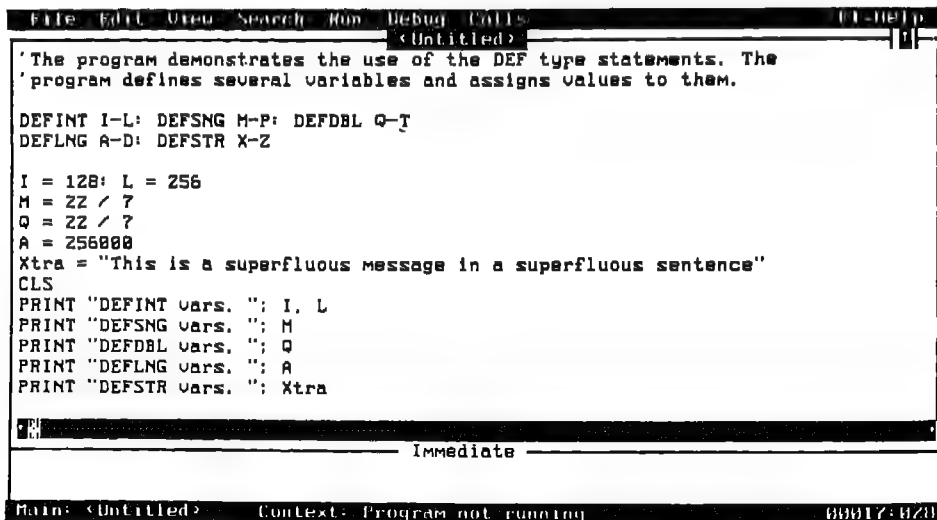
```
DEFSNG P-T: DEFINT X-Z: DEFSTR A-F
..
X = 23: FirstMessage = "Press any key to continue "
```

لاحظ أن المتغير FirstMessage فى المثال لا يتطلب الحرف \$ لتوضيح أنه من النوع الحرفى (السلسلة). فهو له نوع تقليدى من تعريف عبارة DEFSTR.

عملية تقليدية

توضح العملية عبارات DEFtype. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :



```
File Edit View Search Run Debug Calls Help
<Untitled>
The program demonstrates the use of the DEF type statements. The
program defines several variables and assigns values to them.

DEFINT I-L: DEFSNG M-P: DEFDBL Q-T
DEFLNG A-D: DEFSTR X-Z

I = 128: L = 256
M = 22 / 7
Q = 22 / 7
A = 256000
Xtra = "This is a superfluous message in a superfluous sentence"
CLS
PRINT "DEFINT vars. ": I, L
PRINT "DEFSNG vars. ": M
PRINT "DEFDBL vars. ": Q
PRINT "DEFLNG vars. ": A
PRINT "DEFSTR vars. ": Xtra

Immediate

Main: <Untitled> Context: Program not running 00017:020
```

٢ - نفذ البرنامج. لاحظ استخدام عبارات DEFtype في البرنامج.

```
DEFINT vars, 128          256
DEFSNG vars, 3.142857
DEFDBL vars, 3.142857074737549
DEFLNG vars, 256000
DEFSTR vars, This is a superfluous message in a superfluous sentence
```

Press any key to continue

٣ - ارجع إلى البرنامج مع اخلاء الشاشة دون أن تحفظ البرنامج.

٤ - انتقل إلى الدرس الثالث عشر للاستمرار في تسلسل التعلم.

ويمكن لأي منظومة أن تكون استاتيكية \$STATIC أو ديناميكية \$DYNAMIC طبقاً لكيفية توضيحها. والمنظومة الاستاتيكية يتحدد لها ذاكرة عندما يترجم البرنامج. أما المنظومة الديناميكية فلا يكون لها ذاكرة محددة إلا عند التنفيذ، وفيما يلي سرد للطرق التي يمكن أن تكون المنظومة بها استاتيكية أو ديناميكية :

- عندما يوضح ثابت في عبارة CONST أو يستخدم ثابت عددي في عمل أبعاد المنظومة فتكون هذه المنظومة استاتيكية.

- المنظومات التي تحدد لها الأبعاد ضمنياً تكون منظومات استاتيكية.

- المنظومات التي تحدد لها الأبعاد باستخدام متغيرات كأدلة لها تكون منظومات ديناميكية.

ويبلغ أقصى حجم للمنظومة 64 كيلوبايت (65,535 بايت). وعندما تتعدى المنظومة هذا الحد تظهر رسالة خطأ تحدد أن المنظومة أكبر من اللازم، وتوضح إحدى المنظومات بأنها أكبر من 64 كيلوبايت يحدث عن طريق جعلها ديناميكية واستخدام خيار /ah أثناء الترجمة.

ويمكن أن تشمل عبارة البعد بصورة اختيارية الخاصية SHARED. مثل ذلك ما يلي :

```
DIM SHARED variable AS type.
```

وهذا يجعل المتغير متاح الاتصال به عن طريق كل المقاطع، ويمكن أن توضح عبارة البعد أكثر من متغير واحد وذلك باستخدام فواصل تفصلها عن بعضها البعض.

```
DIM Var1 AS STRING, Var2 AS INTEGER, Var3 AS LONG
```

وتضع عبارة البعد قيماً ابتدائية لكل المتغيرات الموضحة، وتكون القيم الابتدائية اصفراً للمتغيرات العددية وفراغات للمتغيرات السلاسل.

وتستخدم عبارة البعد في توضيح متغيرات سجل. ويؤدي ذلك بتعريف نوع السجل بعبارة TYPE و END TYPE ثم توضيح متغير بعد ذلك من هذا النوع، وفيما يلي مثال لذلك :

```
TYPE RecType
Name AS STRING * 30
Address AS STRING * 80
END TYPE
DIM AddressRec AS RecType
```

التطبيقات

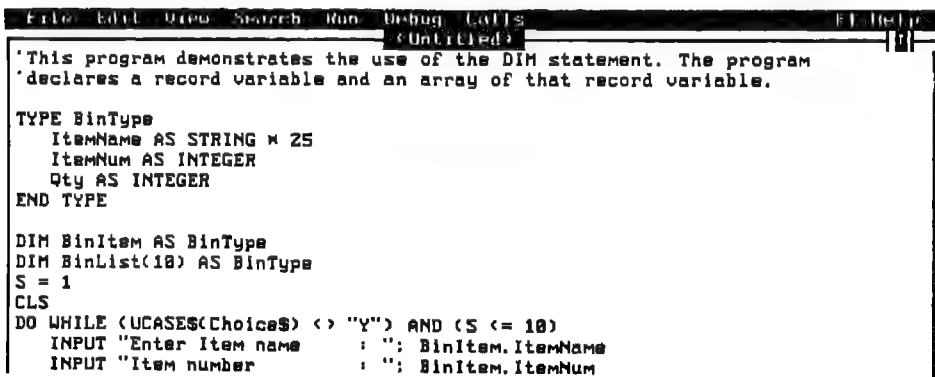
عبارة البعد هي إحدى العبارات الأكثر قوة في توضيح المتغيرات والمنظومات. وتعد الجوانب لعبارة البعد في بيسك السريع يجعلها قابلة للتطبيق في تطبيقات عديدة. فيسمح توضيح المتغيرات التي يعدها المستفيد وتوضيح المتغيرات من النوع البسيط لأن يكون البرنامج أكثر ترتيباً وأسهل في قراءته. وفيما يلي مثال لذلك :

```
DIM A$ AS STRING
DIM A AS INTEGER, B AS LONG, C AS DOUBLE
CONST MaxCol = 80, MaxRow = 25
DIM ScreenArray (MaxRow, MaxCol)
TYPE NewLst
    LastPtr AS STRING * 8
    NextPtr AS STRING * 8
    ThisValue AS INTEGER
END TYPE
DIM CurrNode AS NewLst
```

عملية تقليدية

توضح هذه العملية استخدام عبارة البعد. ابدأ بتحميل بيسك السريع.

١ - أكتب البرنامج التالي :



```
File Edit View Search Run Debug Calls Help
Untitled
This program demonstrates the use of the DIM statement. The program
declares a record variable and an array of that record variable.

TYPE BinType
    ItemName AS STRING * 25
    ItemNum AS INTEGER
    Qty AS INTEGER
END TYPE

DIM BinItem AS BinType
DIM BinList(10) AS BinType
S = 1
CLS
DO WHILE (UCASE$(Choice$) <> "Y") AND (S <= 10)
    INPUT "Enter Item name      : "; BinItem.ItemName
    INPUT "Item number         : "; BinItem.ItemNum
```

```

INPUT "Quantity" : BiniItem.Qty
INPUT "Done ? (Y/N): "; Choice$
BinList(S) = BiniItem
S = S + 1
LOOP
FOR Cnt = 1 TO S
PRINT BinList(Cnt).ItemName, BinList(Cnt).ItemNum, BinList(Cnt).Qty
NEXT

```

Immediate

Main: (Untitled) Context: Program not running 000 05 001

٢- نفذ البرنامج وادخل البيانات التالية. لاحظ استخدام عبارة البعد.

```

Enter Item name : ? Dust bins
Item number : ? 100
Quantity : ? 24
Done ? (Y/N): ? n
Enter Item name : ? Dust Fans
Item number : ? 200
Quantity : ? 20
Done ? (Y/N): ? n
Enter Item name : ? Bingo Cards
Item number : ? 80
Quantity : ? 2000
Done ? (Y/N): ? y
Dust bins : 100 24
Dust Fans : 200 20
Bingo Cards : 80 2000

```

Press any key to continue

٣- ارجع إلى البرنامج. اختر برنامجاً جديداً بون أن تحفظ هذا البرنامج.

٤- انتقل إلى الدرس المائة والخامس عشر للاستمرار في تسلسل التعلم.

الدرس الرابع والثلاثون

عبارة DO LOOP

الوصف

تمثل هذه العبارة أحد تكوينات التحكم في المسار في بيسك السريع، وتتسبب العبارة في تكرار تنفيذ مجموعة عبارات في البرنامج حتى تتحقق شروط معينة أو طالما أن نتيجة شروط معينة تكون صحيحة، ويمكن أن يأخذ تكوين هذه العبارة إحدى الصيغتين التاليتين :

التكوين الأول

```
DO
  statements
LOOP WHILE | UNTIL Boolean expression
```

التكوين الثاني

```
DO WHILE | UNTIL Boolean expression
  statements
LOOP
```

جزء statement في كل من التكوينين عبارة عن عبارات من برنامج بيسك السريع، وتسمى مجموعة مثل هذه العبارات بكتلة (أو مجموعة) العبارات، وتوجد مجموعة العبارات داخل مكونات التحكم مثل WHILE WEND و FOR NEXT و FUNCTION و SELECT CASE و SUB و DEF FN متعددة الأسطر وغيرها، وبالطبع فإن البرنامج يكون مجموعة عبارات كذلك.

التطبيقات

تمثل هذه العبارة وسيلة تحكم متعددة الجوانب في المسار، والتكوينان اللذان سبق وصفهما لهما تأثيرات مختلفة على تنفيذ البرنامج، ففي التكوين الأول تنفذ مجموعة العبارات أول مرة بغض النظر عما إذا تحقق الشرط أم لا، أي إن مجموعة العبارات تنفذ مرة واحدة على الأقل، وأما في التكوين الثاني إذا لم تتحقق شروط تنفيذ الدورة عند البداية فلا تنفذ مجموعة العبارات على الإطلاق، أي إنه في بعض الحالات يمكن أن تهمل مجموعة العبارات، وفيما يلي أمثلة لذلك :

```
DO
  statement 1
  Q = Q + 1
LOOP WHILE (Q < 0)
```

مثال ١

مثال ٢

```
DO WHILE (Q < 10)
    statement 1
    Q = Q + 1
LOOP
```

في المثال الأول تنفذ الدورة مرة واحدة على الأقل بغض النظر عن قيمة Q. وفي المثال التالي له يمكن أن تهمل الدورة كلية إذا ما تعدت قيمة Q المقدار 10. وكل من الصيغتين لها استخداماتها ويعتمد اختيار أي منهما على التطبيق وعلى تكوين البرنامج. استخدام UNTIL بدلاً من WHILE هو موضوع اختيار فقط طبقاً لتكوين تعبير بولياني.

عملية تقليدية

تسترجع في هذه العملية البرنامج الذي سبق حفظه في الدرس السادس والثمانين وعليك أن تلاحظ عبارة DO LOOP. ابدأ بتحميل بيسك السريع.

١ - اختر فتح البرنامج من قائمة File واضغط على Tab للانتقال إلى الدليل واختر الملف LRTRIM. BAS من الدليل مستخدماً مفاتيح الأسهم ثم اضغط على مفتاح الإدخال.

٢ - اضغط على Shift-F2 لتنقيح عبارة FUNCTION. عدل عبارة DO LOOP كما هو مبين في القائمة التالية. احذف كذلك السطر الذي يسبق عبارة DO مباشرة.

```
File Edit View Search Run Debug Calls
LRTRIM.BAS
' This program demonstrates the use of LTRIM$ and RTRIM$ functions
' This program is modified from Module 75

DECLARE FUNCTION GCh$(UCH AS STRING, X AS INTEGER, Y AS INTEGER, P AS STRING)
DIM UCH AS STRING * 30, P AS STRING * 75
CLS
LOCATE 23, 1
PRINT "E>dit / C>reate / D>elete / Q>uit ?"
UCH = "EDCQ"
P = "Enter selection"
Selection$ = GCh$(UCH, 24, 1, P)

File Edit View Search Run Debug Calls
F1-Help
```

```

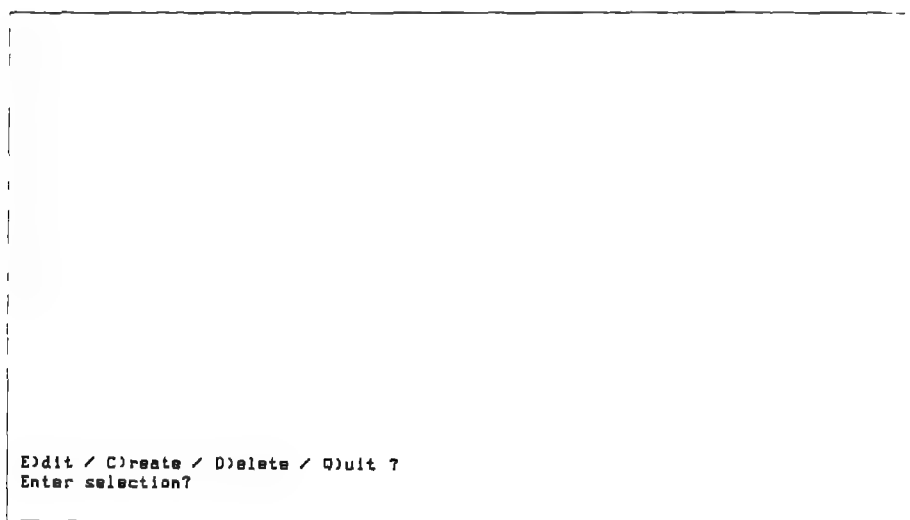
FUNCTION GCh$ (UCh AS STRING, x AS INTEGER, y AS INTEGER, P AS STRING)
  DIM Choice AS STRING
  P = LTRIM$(RTRIM$(P))
  Choice = " "
  DO WHILE (INSTR(UCh, UCASE$(Choice)) = 0)
    LOCATE x, y
    PRINT P:
    INPUT ; Choice
    LOCATE x, (y + LEN(P))
    PRINT Choice:
  LOOP
  GCh$ = Choice
END FUNCTION

```

Immediate

Main: LTRIM.BAS Context: Program not running 00001:001

٣ - نفذ البرنامج ولاحظ استخدام عبارة DO LOOP في البرنامج.



- ٤ - اضغط على أى مفتاح للعودة إلى البرنامج. اختر Save As من قائمة File واكتب DO-LOOP. BAS ثم اكتب اسم الملف واحفظ هذا البرنامج على أنه برنامج نصي.
- ٥ - من قائمة File اختر برنامجاً جديداً واكتب N لإخلاء الشاشة.
- ٦ - انتقل إلى الدرس الثامن للاستمرار في تسلسل التعلم.

الدرس الخامس وثلاثون

عبارة DRAW

الوصف

ترسم هذه العبارة شيئاً على الشاشة طبقاً لما هو مذكور بواسطة تعبير سلسلة المؤشرات، وتكوينها هو كما يلي :

DRAW string exp

جزء string exp هو سلسلة لغة ماكرو يصف كيفية الاستمرار في عمل الرسم، ومكونات لغة الماكرو هي كما يلي :

- أوامر حركة نقطة البداية.
- بادئات لحركة نقطة البداية.
- أوامر معامل مقياس وزاوية دوران وزاوية تحويل ولون للرسم وأوامر للألوان.
- أوامر تنفيذ سلاسل جزئية.

أوامر حركة نقطة البداية هي كما يلي :

الوصف	الجزء
ينقل نقطة البداية عدد n من الوحدات، والوحدة هي نقطة رسم إلا إذا ما تحدد غير ذلك عند وضع أمر معامل المقياس.	$U(n)$
ينقل نقطة البداية عدد n من الوحدات لأسفل.	$D(n)$
ينقل نقطة البداية عدد n من الوحدات لليسار.	$L(n)$
ينقل نقطة البداية عدد n من الوحدات لليمين.	$R(n)$
ينقل من الركن السفلى الأيسر إلى الركن العلوى الأيمن عدد n من الوحدات.	$E(n)$
ينقل من الركن العلوى الأيسر إلى الركن السفلى الأيمن عدد n من الوحدات.	$F(n)$
ينقل من الركن العلوى الأيمن إلى الركن السفلى الأيسر عدد n من الوحدات.	$G(n)$
ينقل من الركن السفلى الأيمن إلى الركن العلوى الأيسر عدد n من الوحدات.	$H(n)$
ينتقل إلى نقطة مطلقة أو نسبية إحداثياتها x, y على الشاشة، عندما يسبق x إشارة فتكون الحركة نسبية للموضع الحالى وإلا فإنها تكون مطلقة، ويرسم خط من الموقع الحالى إلى x, y .	$M\ x, y$

بادئات حركة نقطة البداية

فيما يلي الأوامر التي يمكن استخدامها كبادئات لأي أمر من أوامر الحركة.

البادئة	الوصف
B	ينقل نقطة البداية ولا يرسم أى نقاط
N	ينقل ويرسم إلا أنه يعود إلى الموقع الأصلي.

أوامر TURN و ROTATE و SCALE و COLOR :

تتعامل الأوامر التالية مع زاوية الدوران وزاوية التحويل ومعامل التكبير أو التصغير (المقياس) واختيار اللون فى عبارة DRAW.

الجزء	الوصف
TA n	يتسبب فى دوران الرسم بزاوية مقدارها n درجة حيث تقع n بين 360- و 360. يستخدم أمر TA مع دالة VARPTR\$ مثل : + "TA=" DRAW VARPTR\$ (Angle)
A n	يحدد زاوية دوران الرسم. تقع n بين 0 و 3 حيث 0 يعنى درجة 0 و 1 يعنى 90 درجة و 2 يعنى 180 درجة و 3 يعنى 270 درجة.
S n	يحدد معامل المقياس حيث تقع n بين 1 و 255. ويحدد ذلك حجم وحدات الحركة.
C n	تحدد أن اللون هو n. أرجع إلى الدروس 29 و 98 و 124 لمناقشة الألوان والأعداد.
P n1, n2	تحدد اللون لتلوين الشكل من داخله، n1 هو اللون المستخدم فى التلوين و n2 هو لون الحدود.
V	تجعل كل أوامر DRAW ترسم باستخدام المنطق بدلاً من الاحداثيات الواقعية.

تنفيذ السلاسل الجزئية

تسمح عبارة DRAW بتنفيذ سلاسل أخرى من داخل السلسلة التي تنفذ حالياً، والتكوين هو كما يلي :

```
"X" + VARPTR$(string)
```

من الممكن كذلك تنفيذ سلاسل جزئية أخرى.

التطبيقات

تمثل هذه العبارة وسيلة قوية تنفذ لغة ماكرو لرسم أشياء على الشاشة، ويمكن أن تستخدم عبارة DRAW إذا ما كانت لديك امكانيات رسومات وعرض ملونة فقط، وفيما يلي أمثلة لعبارة : DRAW

مثال ١

```
SCREEN 1
DRAW "E20 F20 E20 F20"
```

مثال ٢

```
Box$ = "U40 R40 D40 L40 E20 P1,1"
DRAW Box$
```

مثال ٣

```
DRAW "M 100,100" + Box$
```

مثال ٤

```
DRAW "M -100,100" + Box$
```

عملية تقليدية

هذه العملية عبارة عن توضيح بسيط لعبارة DRAW، استمر فيها إذا كانت لديك امكانيات رسومات ملونة فقط، ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls F1=Help
DRAW.BAS
' This program demonstrates the use of the DRAW statement.

SCREEN 1
DRAW "c1 u5 r5 d5 15 "
DRAW "bm+20,0 u5 r5 d5 15 "
DRAW "bm-8,0 d10"
DRAW "bm-5,5 r10"
DRAW "bm+15,0 u30 l35"
xp = POINT(0): yp = POINT(1)
DRAW "d30 f10 r15 a10"
DRAW "bm=" + UARPTR$(xp) + ",=" + UARPTR$(yp)

FOR Cnt = 1 TO 12
  DRAW "c2 n85 "
  DRAW "bm+3,0"
NEXT

```

Immediate

Data: DRAW.BAS Context: Program not running 00000-020

٢ - نفذ البرنامج، لاحظ استخدام عبارة DRAW في البرنامج.

٣ - ارجع إلى البرنامج واحفظه كملف نصي تحت اسم ملف DRAW.BAS.

٤ - انتقل إلى الدرس السادس عشر للاستمرار في تسلسل التعلم.

الدرس السادس والثلاثون

عبارة END

الوصف

تستخدم عبارة END فى انتهاء برنامج البيسك أو فى انتهاء إجراء أو فى انتهاء مجموعة من مجموعات البيسك، وتكوين انتهاء كل منها هو كما يلى :

END DEF | FUNCTION | SELECT | IF | SUB | TYPE

الوصف	الجزء
كلمة من كلمات البيسك المحجوزة، واستخدامها فى حد ذاتها يحدد نهاية برنامج البيسك وتقوم باغلاق كل الملفات وتعود إلى بيئة التشغيل.	END
كلمة من كلمات البيسك المحجوزة تستخدم مع END فى انتهاء عبارة DEF FN.	DEF
كلمة من كلمات البيسك المحجوزة تستخدم مع END فى انتهاء إجراء دالة .FUNCTION	FUNCTION
كلمة من كلمات البيسك المحجوزة تستخدم مع END فى انتهاء مكون SELECT .CASE	SELECT
كلمة من كلمات البيسك المحجوزة تستخدم مع END فى انتهاء مكون IF.. THEN.. على هيئة مجموعة.	IF
كلمة من كلمات البيسك المحجوزة تستخدم مع END فى انتهاء إجراء SUB.	SUB
كلمة من كلمات البيسك المحجوزة تستخدم مع END فى انتهاء عبارة تعريف أنواع يحددها المستفيد.	TYPE

التطبيقات

يمكن استخدام عبارة END فى انتهاء البرنامج من أى موقع فى البرنامج. فإذا حذفت كلمة END فيفتراض مترجم البيسك السريع أن البرنامج ينتهى عندما لا تكون هناك أسطر أخرى فى البرنامج لتنفيذها، وفيما يلى بعض الأمثلة :

مثال ١

```
'Program to print 10 random numbers
RANDOMIZE TIMER      ' seed the random number generator with the timer value
FOR i = 1 TO 10
    PRINT INT((20-0+1) * RND+0) " ":
NEXT i
END      ' of program
```

يبين ذلك استخدام END فى انهاء البرنامج. يمكن أن تحذف عبارة END دون أى تأثير على تنفيذ البرنامج.

مثال ٢

```
DEF FNTokenize%(InpStr$)
    FNTokenize% = 0
    FOR I = 1 to LEN(InpStr$)
        Token% = Token% + ASC(MID$(InpStr$,I,1))
    NEXT I
    FNTokenize% = Token%
END DEF
```

يبين هذا المثال تعريف دالة العودة إلى token التى تنتج بواسطة اضافة قيم ASCII لكل الرموز الموجودة فى المؤشر. تستخدم عبارة END فى انهاء التعريف.

مثال ٣

```
'demonstrates the SELECT CASE statement
PRINT "Master / Transaction / Report / Main menu ?"
INPUT "Enter choice ", C$
SELECT CASE C$
    CASE "M"
        GOSUB MasterFileProcess
    CASE "T"
        GOSUB TransFileProcess
    CASE "R"
        GOSUB ReportProcess
    CASE "A"
        GOTO RetMenu2
    CASE ELSE
        PRINT "Invalid selection ..."
END SELECT
```

يبين هذا المثال كيفية استخدام عبارة SELECT CASE فى بناء تكوين متعدد الفروع واستخدام END CASE فى انتهاء التكوين.

مثال

```
FUNCTION LowerCase$ (InpStr$)
  FOR I = 1 TO LEN(InpStr$)
    IF INSTR ("ABCDEFGHIJKLMNOPQRSTUVWXYZ", MID$(InpStr$,I,1)) THEN
      MID$(InpStr$,I) = CHR$(ASC(MID$(InpStr$,I,1)+32))
    END IF
  NEXT I
  LowerCase$ = InpStr$
END FUNCTION
```

يستخدم هذا المثال عبارتى END IF و END FUNCTION. وتحول الدالة سلسلة إلى حروف صغيرة.

مثاله

```
SUB . DisplayPrompt(PromptStr$)
  LOCATE 24,1: PRINT PromptStr$;
END SUB
```

هذا مثال لاستخدام .END SUB

مثال

```
TYPE WindowParms
  WindowData AS STRING[4000]
  LastX      AS INTEGER
  LastY      AS INTEGER
END TYPE
```

يبين هذا المثال استخدام .END TYPE

عملية تقليدية

يوضح مثال البرنامج التالى استخدام عبارة .END. ابدأ بتحميل ببسك السريع.

١ - اكتب البرنامج التالى :

```

File Edit View Search Run Debug Calls F1=Help
<Untitled>
DECLARE FUNCTION LowerCase$ (InpStr$)
PRINT LowerCase$("ABCDEFGHIJKLMNOPQRSTUVWXYZ")

File Edit View Search Run Debug Calls F1=Help
<Untitled>:LowerCase
FUNCTION LowerCase$ (InpStr$)
  FOR I = 1 TO LEN(InpStr$)
    IF INSTR("ABCDEFGHIJKLMNOPQRSTUVWXYZ", MID$(InpStr$, I, 1)) THEN
      MID$(InpStr$, I) = CHR$(ASC(MID$(InpStr$, I, 1)) + 32)
    END IF
  NEXT I
  LowerCase$ = InpStr$
END FUNCTION

Immediate
Mode: >Untitled< Context: Program not running 000001-001

```

٢ - اضغط على Shift-F5 لتنفيذ البرنامج. لاحظ استخدام عبارة هذا الفصل في البرنامج.

```

abcdefghijklmnopqrstuvwxyz

Press any key to continue

```

٣ - اضغط على أى مفتاح للعودة إلى البرنامج.

٤ - اختر Save من قائمة File واكتب END.BAS كاسم للملف. حدد أن الملف من النوع النصي واحفظ البرنامج.

٥ - اخل الشاشة عن طريق اختيار برنامج جديد من قائمة File.

٦ - انتقل إلى الدرس المائة والسابع والأربعون للاستمرار في تسلسل التعلم.

الدرس السابع والثلاثون

عبارة ENVIRON ودالة ENVIRON\$

الوصف

عبارة ENVIRON ودالة ENVIRON\$ تتعاملان مع متغيرات بيئة DOS وذلك مثل PATH و PROMPT\$، وتكوينهما هو كما يلي :

```
ENVIRON string expression
ENVIRON$(environment string)
ENVIRON$(n)
```

عبارة ENVIRON: جزء expression هو السلسلة التي يراد عملها كجزء من بيئة DOS. ويشمل تعبير السلسلة متغير البيئة والقيمة الجديدة التي تحدد له، ويوجد متغير البيئة بالفعل فإذا لم يكن موجوداً فإنه يضاف إلى القائمة، وتعبير السلسلة له الصيغة التالية :

```
parameter=text
```

جزء parameter هو متغير بيئة كما أن جزء text هو القيمة الجديدة المحددة لهذا المتغير.

دالة ENVIRON\$: تعيد كلاً من صيغتي التكوين لدالة ENVIRON\$ السلسلة الحالية المصاحبة لمتغير البيئة المطلوب، عندما تستخدم سلسلة البيئة فتكون السلسلة هي المؤشر المطلوب له معلومات، عندما تستخدم n فنقرأ قائمة متغيرات البيئة ويعود المؤشر رقم n والنص المصاحب له، مثال ذلك تعطى ENVIRON\$ ("PATH") أعداد المسار الحالى وتعطى (1) ENVIRON\$ أعداد أول متغير بيئة.

التطبيقات

تستخدم عبارة ENVIRON ودالة ENVIRON\$ فى الاتصال بمتغيرات بيئة DOS وتعديلها، والاستخدام محدد بمتطلبات التطبيق، وفيما يلي أمثلة لذلك :

مثال ١

```
ENVIRON "PATH=C:\;C:\DOS"
```

مثال ٢

```
PRINT ENVIRON$("PATH")
```

مثال ٣

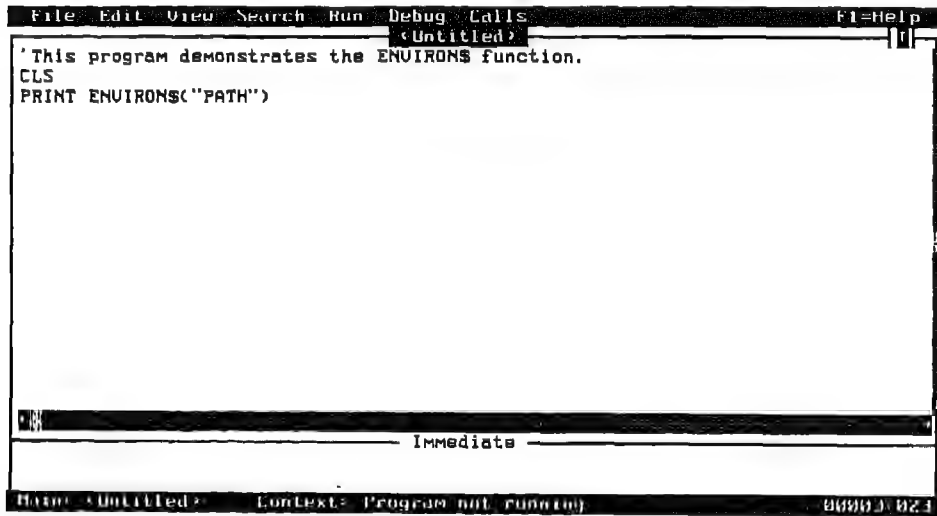
PRINT ENVIRON\$(10)

عندما يوجد المؤشر في قائمة متغير البيئة فتعيد ENVIRON\$ سلسلة فارغة. ويظل تأثير عبارة ENVIRON أثناء تنفيذ البرنامج فقط وبمجرد انتهاء البرنامج تعود متغيرات بيئة DOS مرة أخرى إلى المكان الذي كانت فيه من قبل.

عملية تقليدية

تمثل هذه العملية توضيحاً بسيطاً لدالة ENVIRON\$. ابدأ بتحميل ببسك السريع.

١ - اكتب البرنامج التالي :



٢ - لاحظ استخدام دالة ENVIRON\$ في البرنامج. ارجع إلى البرنامج داخل الشاشة لـ أن تحفظ البرنامج.

٣ - انتقل إلى الدرس السابع عشر للاستمرار في تسلسل التعلم.

الدرس الثامن والثلاثون

دالة EOF

الوصف

تستخدم دالة EOF في التأكد من شرط انتهاء الملف. وتعيد قيمة عددية صحيحة. وتكوينها هو كما يلي :

EOF(filename)

جزء filename هو الرقم المحدد للملف في عبارة OPEN. وتحدد القيمة 1- بأنها صحيحة (وهي نهاية الملف المصاحب لرقم الملف)، كما تحدد أى قيمة أخرى بأنها خاطئة لاختبار نهاية الملف. فإذا ما كانت هناك محاولة لإدخال بيانات بعد الوصول إلى نهاية الملف فتظهر رسالة خطأ بأنه لا إدخال لبيانات.

وعند الاستخدام مع وحدة اتصالات فتعيد دالة EOF نتيجة شرط انتهاء الملف على النحو التالي:

ASCII FILE : تصبح نهاية الملف صحيحة في هذه الحالة عندما يتم استقبال Ctrl-Z وطالما أنه صحيح وحتى يفلق الملف.

BINARY FILE : تصبح نهاية الملف صحيحة في هذه الحالة عندما يكون صف المدخلات (Loc(filename) = 0).

ولا يمكن استخدام EOF على الوحدات التالية : SCRN و KYBRD و CONS و LPTx، حيث x هو رقم صحيح.

التطبيقات

إستخدم دالة EOF كلما كان مطلوباً عمل اتصال بالملف. ويمنع ذلك من القراءة بعد الملف وحدوث خطأ وقت التشغيل في البرنامج وفيما يلي بعض الأمثلة.

مثال ١

```
OPEN "Sales.Dat" FOR INPUT AS #3
WHILE NOT EOF(3)
  ::
WEND
```

مثال ٢

```
OPEN "Sess.Log" FOR RANDOM AS #1
..
IF EOF(1) THEN
ELSE ..
..
```

عملية تقليدية

توضح هذه العملية استخدام دالة EOF، ابدأ بتحميل بيك السريع.

١ - اكتب البرنامج التالي :

```
CLS
OPEN "Print.Fil" FOR INPUT AS #3
CLS
DO WHILE NOT EOF(3)
    LINE INPUT #3, AS
    PRINT AS
LOOP
CLOSE #3
```

Immediate

Main: <Untitled> Context: Program not running 00012:0001

٢ - نفذ البرنامج، لاحظ استخدام دالة EOF في البرنامج.

Earth Moving Equipment	12	\$120,000
Farm Equipment	22	\$85,000
Farm ZZ	\$85.0	
SOS		
Press any key to continue		

٣ - اختر برنامجاً جديداً من قائمة File واختر عدم حفظ البرنامج.

٤ - انتقل إلى الدرس الثامن والسبعين للاستمرار في تسلسل التعلم.

الدرس التاسع والثلاثون

عبارة ERASE

الوصف

تلفى عبارة ERASE مواقع الذاكرة المحددة لمنظومة ديناميكية أو تعيد وضع القيم الابتدائية لعناصر منظومة استاتيكية وتكوينها هو كما يلي :

ERASE array

جزء array هو المنظومة التي تتأثر. فإذا كانت المنظومة ديناميكية فتخلي الذاكرة المستخدمة بواسطة المنظومة، أما إذا كانت المنظومة استاتيكية فتوضع قيم ابتدائية لعناصرها. عندما تكون المنظومة استاتيكية فتوضع قيم صفر للعناصر العددية وقيم فراغات لعناصر السلاسل. يمكن وضع قيم ابتدائية لأكثر من منظومة واحدة كما يمكن الغاء مواقع أكثر من منظومة واحدة وذلك عن طريق استخدام فواصل تفصل بين المنظومات فى عبارة ERASE مثل ما يلي :

ERASE Array1, Array2, Array3.

التطبيقات

عبارة ERASE هى وسيلة أخرى لإدارة ذاكرة وقت التنفيذ فى بيسك السريع. ويمكن استخدام عبارة REDIM مع المنظومات الديناميكية بدون استخدام عبارة ERASE وذلك لأن عبارة REDIM ترعى إعادة عمل الأبعاد للمنظومة الديناميكية. وفيما يلي بعض الأمثلة :

```
'$STATIC
DIM Art(320,680)
...
ERASE Art
```

يوضح المثال السابق عبارة ERASE المستخدمة فى إعادة وضع قيم ابتدائية فى المنظومة الاستاتيكية Art.

```
'$DYNAMIC
DIM PictureArr(225,420)
...
REDIM PictureArr(320,680)
...
ERASE PictureArr
```

يوضح المثال السابق استخدام REDIM بدون تدخل من ERASE واستخدام ERASE في الغاء الذاكرة المحددة لـ PicturArr.

عملية تقليدية

توضح هذه العملية استخدام عبارة ERASE. ابدأ بتحميل بيك السريع.

١ - اكتب البرنامج التالي :

```
File Edit View Search Run Debug Lists F1-Help
Untitled
' This program demonstrates the use of the ERASE statement. The program
' uses a dynamic array to find the minimum and maximum values of the two
' lists. The ERASE statement is used to get rid of the array
' before redimensioning.

Max = 15
DIM A(Max)
GOTO Start

LoadArray:
  FOR Cnt = 1 TO Max
    READ A(Cnt)
  NEXT
  RETURN

FindMinMax:
  MinVal = A(1): MaxVal = A(1)
  FOR Cnt = 2 TO Max
    IF MinVal > A(Cnt) THEN
      MinVal = A(Cnt)
    END IF
    IF MaxVal < A(Cnt) THEN
      MaxVal = A(Cnt)
    END IF
  NEXT
  RETURN

Start:
  GOSUB LoadArray
  GOSUB FindMinMax
  PRINT "First pass"
  PRINT "Minimum of array: "; MinVal, "Maximum of array: "; MaxVal
  READ Max
  ERASE A
  DIM A(Max)
  GOSUB LoadArray
  GOSUB FindMinMax
  PRINT "Second pass"
  PRINT "Minimum of array: "; MinVal, "Maximum of array: "; MaxVal
  ERASE A
```

```
DATA 12,23,33,43,1,56,98,656,323,44,9,88,67,54,18
DATA 18
DATA 8,89,76,54,23,32,12,4,33,54
```

Immediate

DATA: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

٢ - نفذ البرنامج. لاحظ استخدام ERASE في البرنامج. أعد وضع أبعاد المنظومة بدون استخدام عبارة REDIM.

```
First pass
Minimum of array: 1      Maximum of array: 656
Second pass
Minimum of array: 4      Maximum of array: 89
```

Press any key to continue

٣ - ارجع إلى البرنامج واخل الشاشة دون أن تحفظ هذا البرنامج.

٤ - انتقل إلى الدرس الثاني والثلاثين للاستمرار في تسلسل التعلم.

الدرس الأربعون

دالتا ERDEV\$ و ERDEV

الوصف

تستخدم دالتا ERDEV و ERDEV\$ في الحصول على حالة خطأ محددة في الوحدة. وتكونيهما هو كما يلي :

```
ERDEV
ERDEV$
```

ولا تحتاج أى من الدالتين لوجود مؤشر ولا يقبل أى مؤشر. وتعطى دالة ERDEV رمز الخطأ للوحدة التى انتجت الخطأ. كما تعطى دالة ERDEV\$ اسم الوحدة التى حدث عندها الخطأ. والقيمة التى تعود من دالة ERDEV تكون على هيئة شفرة بت مع ضغط أدنى ثمانية بت برمز الخطأ في DOS. ويوجد المزيد من فك قيم شفرة البت في الدليل التقنى لنظام DOS.

التطبيقات

تستخدم دالتا ERDEV و ERDEV\$ في الحصول على معلومات عن شروط الخطأ في الوحدات مثل مشغل الأقراص أو بوابات الاتصالات. وفيما يلي أمثلة لاستخدام ERDEV و ERDEV\$.

مثال ١

```
ON ERROR GOTO PrintErrAndStop
..
PrintErrAndStop:
PRINT "Error code on ";ERDEV$;" is ";ERDEV STOP
```

مثال ٢

```
ON ERROR GOSUB SaveError
..
SaveError:
DevError = ERDEV: DevName = ERDEV$
..
```

عملية تقليدية :

توضح هذه العملية استخدام دالتي ERDEV و ERDEV\$. يحاول البرنامج أن يعرض الدليل في المشغل A ويظهر الخطأ لأن باب مشغل الأقراص يكون مفتوحاً. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls F1=Help
<Untitled>
' This program demonstrates the ERDEV and ERDEV$ functions.
ON ERROR GOTO PrintErrAndStop

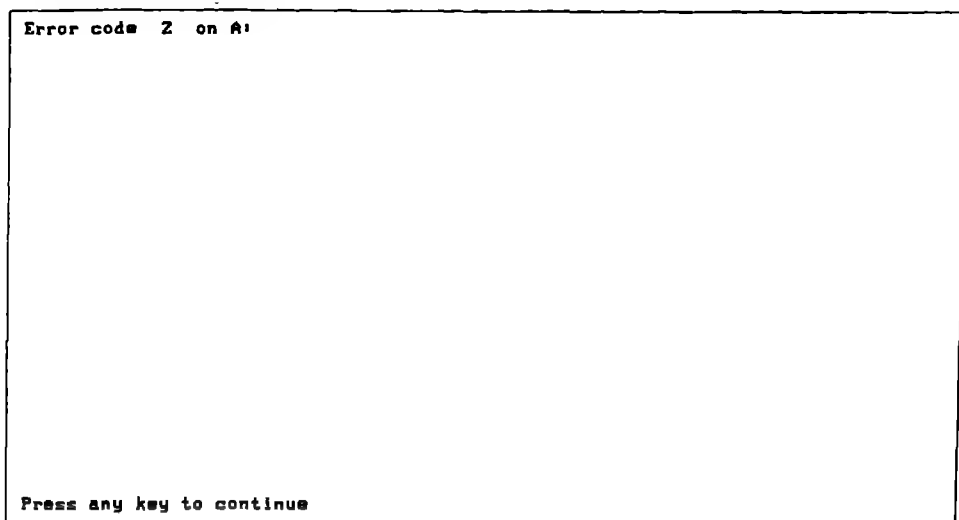
CLS
FILES "A:\*.Bas"

PrintErrAndStop:
PRINT "Error code "; ERDEV: " on "; ERDEV$
END

----- Immediate -----
Main: <Untitled> Context: Program not running 00005:006

```

٢ - نفذ البرنامج مع ترك باب مشغل الأقراص A مفتوحاً. نتيجة البرنامج هي كما يلي. لاحظ استخدام دالتي ERDEV و ERDEV\$ في البرنامج.



- ٣ - ارجع إلى البرنامج مع اخلاء الشاشة نون أن تحفظ البرنامج.
- ٤ - انتقل إلى الدرس التاسع والخمسين للاستمرار في تسلسل التعلم.

الدرس الحادى والأربعون

دالة ERR و ERL وعبارة ON ERROR GOTO

الوصف

دالة ERR : تعيد دالة ERR رمز الخطأ لحدث خطأ وقع فى البرنامج. وعندما تختبر للحصول على النتائج الفورية بعد الخطأ فتعيد دالة ERR رمز الخطأ المصاحب لهذا الخطأ. وتكوينها هو كما يلي :

ERR

ولا يوجد مؤشر لدالة ERR. فهي تعيد قيمة صحيحة تقع بين 1 و 255. لا يمكن استخدام دالة ERR فى الطرف الأيسر من عبارة التحديد.

دالة ERL : تعيد دالة ERL رقم السطر الذى حدث عنده أحدث خطأ. وتكوينها هو كما يلي:

ERL

ولا يوجد مؤشر لدالة ERL. فإذا لم توجد أى أرقام أسطر فى البرنامج فتعيد دالة ERL الرقم 0. ولاتعيد اسم السطر. ودالة ERL مثل دالة ERR لاتستخدم فى الطرف الأيسر لعبارة التحديد.

عبارة ON ERROR GOTO : تمكن عبارة ON ERROR GOTO من معالجة الخطأ داخل البرنامج. وتكوينها هو كما يلي :

ON ERROR GOTO line

جزء GOTO line من العبارة يحدد رقم أول سطر أو اسم السطر المصاحب لمقطع معالجة الخطأ. وعندما يحدث أحد الأخطاء يقفز التحكم فى البرنامج إلى السطر المحدد ويستمر التنفيذ من عند هذا السطر. ورقم السطر 0 يلغى المقدرة على اصطياذ الخطأ. وعندما تنفذ عبارة ON ERROR GOTO 0 داخل معالج الخطأ فيطبع البرنامج رسالة الخطأ وينتهى تشغيل البرنامج كلية.

ومعالج الخطأ ليس برنامجاً فرعياً. ولاتستخدم توكينات SUB و DEF FN و FUNC- TION كمعالجات للخطأ.

التطبيقات :

تقدم نوال ERR و ERL وعبارة ON ERROR GOTO آلية قوية تسمح باصطياد خطأ وقت التشغيل في البرنامج. وتنفيذ هذه الوسائل بسيط ويمكن على ذلك أن يستخدمه المبتدئون. ويقلل هذا النوع من اصطياد الأخطاء من الحاجة إلى العديد من عبارات GOTO وعبارات IF..THEN..ELSE وعبارات GOSUB المستخدمة في التحكم في تداخل البرنامج مع الاستفادة ومع وحدات المدخلات والمخرجات. وتتولى بيئة وقت تشغيل برنامج بيسك السريع اصطياد الأخطاء المهمة وتسمح بتبسيط شيق للبرنامج. وفيما يلي بعض الأمثلة :

دالة ERR :

مثال ١

```
IF ERR = 53 THEN PRINT "File not found. Please try again."
```

مثال ٢

```
WHILE ERR = 0
  statement
statement 2
WEND
```

مثال ٣

```
NewError = ERR
PRINT NewError
```

دالة ERL :

مثال ١

```
IHandleErrors:
  NewError = ERR
  IF NewError = 12 THEN PRINT NewError;" at ";ERL
END
```


مثال ٢

```
NewErrorLine = ERL
WRITE #2 "Error occurred at .";NewErrorLine
```

عبارة ON ERROR GOTO

مثال ١

```
'Program to print the names and addresses
'All global variables, no subroutines

DEFINT A-G
ON ERROR GOTO 1000
...
1000 PRINT ERR,"Error occurred. What is going on?"
END
```

مثال ٢

```
ON ERROR GOTO CrashHelmet
...
CrashHelmet:
...
RESUME
```

عملية تقليدية :

توضح هذه العملية استخدام دالة ERR وعبارة ON ERROR GOTO ودالة ERL. ابدأ
بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls F1=Help
<Untitled>
'This is a demonstration of the ERR and ON ERROR GOTO statements.
'The program attempts to display files from the logged directory.
'The error code 53 means "file not found"; translated in this context,
'it means the file specification is invalid.

ON ERROR GOTO ErrorHandler
CLS
PRINT : PRINT "List of files with the extension PAS "
FILES "M.PAS"
END
ErrorHandler:
  IF ERR = 53 THEN
    PRINT "File specification is invalid ..."
  ELSE
    PRINT "Don't know what happened .."
  END IF
END

```

Immediate

Main: <Untitled> Context: Program not running 00001:001

٢ - نفذ البرنامج، لاحظ استخدام ERR و ERL و ON ERROR GOTO في البرنامج.
اضغط على أى مفتاح للعودة للبرنامج.

```

List of files with the extension PAS
C:\QB
File specification is invalid ...

Press any key to continue

```

٣ - غير في البرنامج عن طريق تغيير PAS إلى BAS. ارجع إلى البرنامج ملاحظاً أن الأخطاء قد تم تصحيحها.

```

List of files with the extension BAS
C:\QB
SAMPLE .BAS      REMLINE .BAS      SORTDEMO.BAS      TORUS .BAS
DEMO1 .BAS       DEMO2 .BAS       DEMO3 .BAS       INCHZCM .BAS
BOX .BAS         BOXZ .BAS        PRINT .BAS        STRING .BAS
IFTHEN .BAS      CASE .BAS        ASC .BAS          ULCASE .BAS
LTRIM .BAS       ABS .BAS         RANDOM .BAS       DOLOOP .BAS
PLAY .BAS        ONEVENT .BAS
898888 Bytes free

```

Press any key to continue

٤ - اضغط على أى مفتاح للعودة إلى البرنامج، اختر New من قائمة File واكتب N لإخلاء الشاشة.

٥ - انتقل إلى الدرس الثانى والأربعين للاستمرار فى تسلسل التعلم.

الدرس الثانى والأربعون

عبارة ERROR

الوصف

تسمح عبارة ERROR لك بتعريف رموز خاصة بك للأخطاء أو بمحاكاة خطأ ببسك عندما يستدعى برمز خطأ موجود بالفعل. وتكوينها هو كما يلي :

ERROR integer expression

جزء integer expression تقع قيمته بين 1 و 255. تحاكي عبارة ERROR شرط الخطأ المحدد عندما يكون التعبير العددي الصحيح عبارة عن رمز خطأ موجود. وإلا فإن عبارة ER-ROR تعرف التعبير العددي الصحيح كرمز جديد للخطأ. وتعيد دالة ERROR رموز الخطأ سابقة التعريف والتي يعرفها المستفيد. وأحدى الطرق الجيدة لتعريف رموز خطأ خاصة بك هي البدء من 255 والعمل لأسفل وهذا يسمح بالتوافقية مع صيغ ببسك السريع المستقبلية التي يمكن أن تكون بها رموز جديدة للخطأ معرفة داخلها.

التطبيقات :

عبارة ERROR هي حليف قوى مع الآليات الأخرى لمعالجة الأخطاء والمتوفرة في ببسك II سريع. المقدرة على تعريف رموز خطأ جديدة محددة للتطبيق هي إحدى طرق التعود على بيئة وقت التنفيذ. وفيما يلي بعض الأمثلة :

مثال ١

ERROR 12 'simulates the error code 12

مثال ٢

ERROR 53 'simulates the error code 53

مثال ٣

' assigns new error code 255 that can be defined in the
' error handling routine

عملية تقليدية :

توضح هذه العملية استخدام عبارة ERROR. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls F1=Help
<Untitled>
' This is a demonstration of the ERROR statement. The program uses the
' ERROR statement to define new error conditions and to trap occurrences
' of that error.

ON ERROR GOTO Eh
CLS : PRINT
PRINT : PRINT "This program demonstrates the ERROR statement "
INPUT "Do you wish to continue (Y/N) "; Yn$
IF Yn$ = "Y" OR Yn$ = "y" THEN
    ERROR Z55
ELSE
    IF Yn$ = "N" OR Yn$ = "n" THEN
        ERROR Z54
    ELSE ERROR Z53
    END IF
END IF
END

Eh:
IF ERR = Z55 THEN
    PRINT "What for? There is nothing going on here ..."
ELSE
    IF ERR = Z54 THEN
        PRINT "Why not? You don't know what you are missing!"
    ELSE PRINT "What? Don't you want to play along ..."
    END IF
END IF
END

Immediate

Main: <Untitled> Context: Program not running 88836-883

```

٢ - نفذ البرنامج. لاحظ استخدام عبارة ERROR في البرنامج. اكتب Y واضغط على مفتاح الادخال. اضغط على أى مفتاح للعودة إلى البرنامج.

```
This program demonstrates the ERROR statement
Do you wish to continue (Y/N)  ? y
What for? There is nothing going on here ...
```

Press any key to continue

٢ - اختر New من قائمة File وأكتب N لاختلاء الشاشة.

٤ - انتقل إلى الدرس الرابع والتسعين للاستمرار في تسلسل التعلم.

الدرس الثالث والأربعون

عبارة EXIT

الوصف :

تستخدم عبارة EXIT في ترك دورة أو دالة أو برنامج فرعى تركاً نهائياً، وتكوينها هو كمايلي:

EXIT structure

جزء structure هو أى تكوين في بيسك السريع، وفيما يلي قائمة بصيغ مختلفة لعبارة EXIT:

```
EXIT FOR
EXIT LOOP
EXIT DEF
EXIT FUNCTION
EXIT SUB
```

كل من صيغ عبارة EXIT يناسب غرضاً محدداً فقط مذكوراً على النحو التالي. واستخدام هذه الصيغ دون تمييز لغرض استخدام كل منها يتسبب في خطأ وقت الترجمة.

عبارة EXIT FOR : تتسبب هذه العبارة في الخروج من دورة FOR، ويستمر تنفيذ البرنامج من السطر الذي يلي دورة FOR مباشرة، وعندما تكون دورات FOR متداخلة فيستمر تنفيذ البرنامج من عند السطر الذي يلي دورة FOR الحالية مباشرة.

عبارة EXIT LOOP : تتسبب هذه العبارة في الخروج من دورة DO LOOP، ويستمر تنفيذ البرنامج من السطر الذي يلي دورة DO LOOP مباشرة، وعندما تكون دورات DO LOOP متداخلة فيستمر تنفيذ البرنامج من عند السطر الذي يلي دورة DO LOOP الحالية مباشرة.

عبارة EXIT DEF : تتسبب هذه العبارة في الخروج من دالة DEF، ويستمر تنفيذ البرنامج من عند السطر الذي استدعى DEF FN.

عبارة EXIT FUNCTION : تتسبب هذه العبارة في الخروج من دالة FUNCTION، ويستمر تنفيذ البرنامج من عند السطر الذي استدعى FUNCTION.

عبارة EXIT SUB : تتسبب هذه العبارة في الخروج من برنامج فرعى SUB. ويستمر تنفيذ البرنامج من عند السطر الذي استدعى SUB.

التطبيقات :

توفر دالة EXIT مرونة في ترك التكوين تركاً نهائياً عندما تؤدي الشروط إلى ذلك. مثال ذلك تنتهي دورة FOR عندما يحصل عداد الدورة إلى الحد المحدد في عبارة FOR فإذا لم يكن هذا بسبب عبارة EXIT FOR فيجبر البرنامج على الاستمرار في تنفيذ الدورة حتى عندما يتحقق شرط معين ومثل هذا الاستمرار غير مناسب. وفيما يلي بعض الأمثلة :

مثال ١

```
FOR Tc = 1 TO 25
  IF (Mens > 0) THEN EXIT FOR
NEXT
```

مثال ٢

```
DO WHILE NoPrint
  IF OutPaper THEN EXIT LOOP
LOOP
```

مثال ٣

```
DEF FNReName(NewName AS STRING)
  IF FileNotFound THEN EXIT DEF
END DEF
```

عملية تقليدية :

توضح هذه العملية استخدام EXIT. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :


```

File Edit View Search Run Debug Calls F1=Help
<Untitled>
This program demonstrates the use of the EXIT statement. The program
generates the period of a pendulum given the length, and exits
the function when the length is less than or equal to 0.

CONST pi = 3.14159
GOTO Start

DEF FnPeriod! (Length)
  IF Length <= 0 THEN EXIT DEF
  FnPeriod! = (2 * pi) * SQR(Length / 981)
END DEF

Start:
CLS
INPUT "Enter the length of the pendulum in centimeters": Length
PRINT "The period is ": FnPeriod!(Length)

Immediate

Main: <Untitled> Context: Program not running 00016-044

```

٢ - نفذ البرنامج. اكتب 1- واضغط على مفتاح الإدخال. لاحظ استخدام EXIT في البرنامج.

```

Enter the length of the pendulum in centimeters? -1
The period is 0

Press any key to continue

```

٣ - ارجع إلى البرنامج مع اخلاء الشاشة لكون أن تحفظ هذا البرنامج.

٤ - انتقل إلى الدرس الحادي والثلاثين للاستمرار في تسلسل التعلم.

الدرس الرابع والأربعون

دالة EXP

الوصف

ت حسب دالة EXP الدالة الأسية لتعبير عددي معين، وتكونها هو كما يلي :

EXP(numeric expression)

والقيمة التي تعيدها الدالة هي e (أساس اللوغاريتم الطبيعي) مرفوعاً لقوة التعبير العددي. تكون قيمة التعبير العددي أقل من 88.02969 وعندما يتعدى التعبير العددي هذه القيمة فتظهر رسالة خطأ السريان الزائد "overflow". وتحسب القيمة التي تعود كقيمة لها دقة فردية بصورة تقليدية. وعندما يكون التعبير العددي له دقة مزدوجة فتحسب دالة EXP بدقة مزدوجة كذلك.

التطبيقات

تعيد دالة EXP القوة الأسية المحددة، قوة للأساس e، والاستخدام محدد بالتطبيق المبرمج فقط.

عملية تقليدية

توضح العملية التالية استخدام دالة EXP. ابدأ بتحميل ببسك السريع.

١ - اكتب البرنامج التالي :

```

File Edit View Window Help
EXP.BAS
' This program computes the hyperbolic trigonometric COSH Function.
' It is calculated Cosh x = e^x + e^-x / 2
CLS
INPUT "Enter x "; x
Temp! = EXP(x) + EXP(-x)
PRINT "Cosh "; x; " is "; Temp / 2

Immediate

Date: EXP.BAS  Contact: Program not running  10/10/2004

```

٢ - نفذ البرنامج. اكتب 5 كاستجابة للملقن. لاحظ استخدام دالة EXP في البرنامج. اضغط على أى مفتاح للعودة إلى البرنامج.

```
Enter x ? 5
CosH 5 is 74.20995

Press any key to continue
```

٣ - من قائمة File اختر New واكتب N لإخلاء الشاشة.
٤ - انتقل إلى الدرس الثانى والثمانين للاستمرار فى تسلسل التعلم.

الدرس الخامس والأربعون

عبارة FIELD

الوصف

تحدد عبارة FIELD مواقع للمتغيرات في الذاكرة الاحتياطية ملف الاتصال المباشر. وتكوينها هو كما يلي :

```
FIELD # filenum, field width AS string var.
```

جزء filenum هو رقم يحدد للملف في عبارة OPEN، وجزء field width هو حجم الحقل في السجل. وجزء string var هو الموقع الذي تخزن فيه البيانات قبل كتابتها في الملف أو بعد قراءتها من الملف. ويمكن أن تعطى تعريفات لحقول متعددة مفصولة عن بعضها البعض بواسطة فواصل. ويجب ألا يزيد إجمالي أطوال الحقول عن حجم السجل المحدد في عبارة OPEN. فإذا ما حدث ذلك فتظهر رسالة تفيد بسريان زائد للحقل "FIELD overflow". يمكن تنفيذ عبارات FIELD متعددة في نفس الوقت وتظل نشطة في نفس الوقت كذلك.

لا تستخدم متغير سلسلة معروفاً كحقل في عبارة INPUT إذا ما أردت أن تستخدمه كحقل نظراً لأنه يعطى نتائج غير متوقعة.

التطبيقات

تسمح عبارة FIELD بتحكم إضافي على البيانات المخزنة في ملف وذلك بتعريف مجموعة رموز للمعالجة. وهي أكثر فائدة عند استخدام ملفات غير نصية أي ملفات بها بيانات مختلطة. وفيما يلي بعض الأمثلة :

```
CONST BinID = 10, BinQty = 10, ReorderPt = 5, RecSize = 25
OPEN "Inven.Dat" FOR RANDOM AS #1 LEN = RecSize
FIELD #1 BinID AS BinNum$, BinQty AS QtyOnHand, ReorderPt AS RP
```

تعرف عبارة CONST في المثال السابق أحجام الحقول وتعد عبارة OPEN الملف وتعرف عبارة FIELD محتويات الحقل.

```
CONST FirstN = 30, MidInit = 1, LastName = 30, RecSize = 61
OPEN "Name.Lst" FOR OUTPUT AS #3 LEN = RecSize
FIELD #3 FirstN AS FirstName$, MidInit AS MidInitial$, LastName AS LName$
```

عملية تقليدية :

توضح هذه العملية استخدام عبارة FIELD. ابدأ بتحميل بيسك السريع.

١ - اختر New من قائمة File واكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls F1=Help
<Untitled>
' This is a demo for the FIELD statement. The program accepts input
' into two variables and writes them to the disk under the filename
' SCORE.DAT.

OPEN "Score.Dat" FOR RANDOM AS #3 LEN = 27
FIELD #3, 25 AS StudentName$, 2 AS Score$

CLS
PRINT "Student name": TAB(40); "Student Score": PRINT STRING$(60, "-")
FOR Cnt = 1 TO 10
    LOCATE Cnt + 2, 1: INPUT SName$
    LOCATE Cnt + 2, 40: INPUT SS
    LSET StudentName$ = SName$
    LSET Score$ = MKIS$(SS)
    PUT #3
NEXT
CLOSE #3
END

Immediate

Main: <Untitled> Context: Program not running 000.06:001

```

٢ - نفذ البرنامج. لاحظ استخدام عبارة FIELD في البرنامج في اعداد الحقول الفردية في الملف. اكتب البيانات التالية واضغط على مفتاح الادخال بعد كل ادخال. بعد 10 ادخالات في كل حقل ينتهي البرنامج.

Student name	Student Score
? Mary Jane	? 34
? Judy O.	? 43
? Susie P.	? 55
? QBert	? 54
? Mark S. Man	? 44
? Henry	? 83
? Pipsqueak	? 89
? Nid	? 67
? Sid	? 78
? Pel	? 78

Press any key to continue

٣ - اضغط على أى مفتاح للعودة إلى البرنامج. اختر New من قائمة File واكتب N لإخلاء الشاشة.

٤ - انتقل إلى الدرس الخامس والثمانين للاستمرار فى تسلسل التعلم.

الدرس السادس والأربعون

دالة FILEATTR

الوصف

تعطى دالة FILEATTR معلومات عن ملف مفتوح. وتكوينها هو كما يلي :

FILEATTR(filename, attribute)

جزء filename هو رقم الملف المستخدم فى عبارة OPEN عندما تم فتح هذا الملف. وجزء attribute أما أن يكون 1 أو 2. وعندما يكون 1 فتعيد دالة FILEATTR رقماً محدداً الحالة التى فتح فيها الملف على النحو التالى :

نتيجة الدالة	حالة الملف
1	مدخلات
2	مخرجات
4	عشوائى
8	اضافة
32	ثنائى

وعندما تكون قيمة attribute هى 2 فتعيد دالة FILEATTR رقم معالجة ملف DOS. ورقم معالجة ملف DOS هو رقم داخلى يستخدمه نظام DOS فى تتبع الملف المفتوح.

التطبيقات

تستخدم دالة FILEATTR فى الحصول على معلومات عن ملف ببسك السريع الذى سبق فتحه. والبرنامج الفرعى الذى ليس له اتصال بعبارة OPEN والذى فتح الملف فعلاً يمكنه أن يحصل على معلومات عن الملف ويتخذ قرارات عن كيفية تشغيله. والاستخدامات الأخرى تعتمد على التطبيق نفسه. وفيما يلي أمثلة لاستخدام FILEATTR :

مثال ١

```
OPEN Fil$ FOR RANDOM AS #2
IF FILEATTR(2,1) = 32 THEN GOSUB ProcessBinary
```

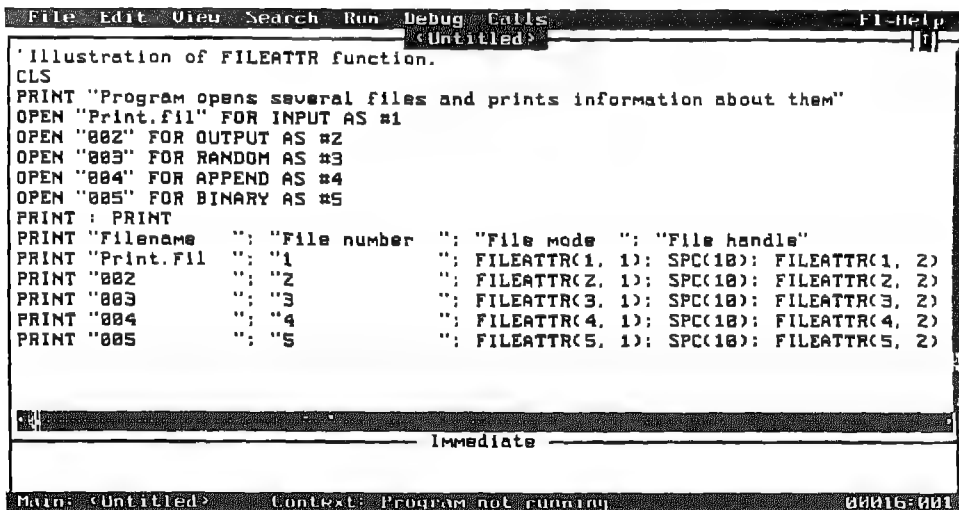
مثال ٢

```
SUB CustomerFileProcess(FNum AS INTEGER)
    FMode = FILEATTR(FNum, 1)
END SUB
```

عملية تطبيقية

تفتح هذه العملية سلسلة من الملفات وتطبع المعلومات التي تأتي من دالة FILEATTR. ابدأ بتحميل ببسك السريع.

١ - اكتب البرنامج التالي :



```
File Edit View Search Run Debug Calls FI-Help
<Untitled>
' Illustration of FILEATTR function.
CLS
PRINT "Program opens several files and prints information about them"
OPEN "Print.fil" FOR INPUT AS #1
OPEN "002" FOR OUTPUT AS #2
OPEN "003" FOR RANDOM AS #3
OPEN "004" FOR APPEND AS #4
OPEN "005" FOR BINARY AS #5
PRINT : PRINT
PRINT "Filename   "; "File number   "; "File mode   "; "File handle"
PRINT "Print.fil   "; "1           "; FILEATTR(1, 1); SPC(10); FILEATTR(1, 2)
PRINT "002         "; "2           "; FILEATTR(2, 1); SPC(10); FILEATTR(2, 2)
PRINT "003         "; "3           "; FILEATTR(3, 1); SPC(10); FILEATTR(3, 2)
PRINT "004         "; "4           "; FILEATTR(4, 1); SPC(10); FILEATTR(4, 2)
PRINT "005         "; "5           "; FILEATTR(5, 1); SPC(10); FILEATTR(5, 2)

Immediate

Main: <Untitled> Context: Program not running 00016:001
```

٢ - نفذ البرنامج. لاحظ استخدام FILEATTR في البرنامج. تكون مخرجات البرنامج كما يلي:


```

Program opens several files and prints information about them

Filename  File number  File mode  File handle
Print.Fil 1          1          5
002       2          2          6
003       3          4          7
004       4          8          8
005       5          32         9

Press any key to continue

```

- ٣ - ارجع إلى البرنامج. اختر New ولا تحفظ هذا البرنامج.
- ٤ - انتقل إلى الدرس الثامن عشر للاستمرار في تسلسل التعلم.

الدرس السابع والأربعون

عبارات FILES و CHDIR و MKDIR و RMDIR

الوصف

عبارة FILES : تستخدم عبارة FILES فى طباعة أسماء الملفات الموجودة على قرص معين أو فى دليل معين، وتكوينها هو كما يلى :

FILES file specification

جزء file specification اختيارى ويذكر لعبارة FILES المكان الذى تبحث فيه عن الملفات ونوع الملفات التى تبحث عنها، وعندما يحذف هذا الجزء فتسرد عبارة FILES كل أسماء الملفات الموجودة على المشغل الحالى أو فى الدليل الحالى. رموز wild-card النمطية فى DOS يمكن استخدامها فى مواصفات الملف، ويجب أن توضع مواصفات الملف بين علامتى تنصيص مزبجتين إذا ما كانت حرفية.

وتخدم عبارة FILES عند استخدامها فى أحد البرامج كوسيلة لجعل البرنامج صديقاً للمستفيد، وفيما يلى بعض الأمثلة لعبارة FILES :

FILES

تطبع هذه العبارة كل الملفات الموجودة على مشغل الأقراص الحالى أو فى الدليل الحالى.

FILES "*.DAT"

تطبع هذه العبارة كل الملفات التى لها اتساع DAT.

FILES "A:*.SCR"

تطبع هذه العبارة كل الملفات الموجودة على مشغل الأقراص : A ولها اتساع SCR..

FILES "C:*.*"

تطبع هذه العبارة كل الملفات الموجودة فى دليل الجذر على القرص : C.

وتطبع عبارة FILES الدليل الحالى كمتوان بغض النظر عن مواصفات الملف.

عبارة CHDIR : تغير عبارة CHDIR الدليل التقليدي على مشغل محدد، وتكوينها هو كما يلي :

CHDIR path specification

جزء path specification هو جزء المسار الجديد للدليل التقليدي. ويمكن أن يكون طوله حتى 64 رمزاً كحد أقصى. وتكوين المسار هو مثل تكوين أسماء مسارات DOS، ويجب أن يوضع وصف الملف بين علامتي تنصيص مزبوجتين إذا كان حرفياً.

تستخدم عبارة CHDIR في تغيير الدلائل أثناء تنفيذ البرنامج، وفيما يلي بعض الأمثلة :

```
CHDIR "\NEW.DAT"
CHDIR "\BASIC\QB.400\MAINT"
```

عبارة MKDIR : تنتج عبارة MKDIR دليلاً جديداً طبقاً لمواصفات المسار المعطاة وتكوينها هو كما يلي :

MKDIR path specification

يقدم جزء path specification اسم الدليل الجديد، ويمكن أن يكون طوله حتى 128 حرفاً كحد أقصى. وتكون مواصفات المسار متطابقة مع تكوين أسماء المسار في DOS، ويجب أن توضع مواصفات الملف بين علامتي تنصيص مزبوجتين إذا كان حرفياً.

تستخدم عبارة MKDIR في انتاج أدلة جديدة أثناء تنفيذ البرنامج، وفيما يلي بعض الأمثلة:

```
MKDIR "C:\BASIC"
MKDIR "C:\PASCAL"
MKDIR "\TEMP"
```

عبارة RMDIR : تحذف عبارة RMDIR دليلاً طبقاً للمسار المحدد، وتكوينها هو كما يلي:

RMDIR path specification

جزء path specification هو للدليل المراد حذفه، ويمكن أن يكون طوله حتى 128 حرفاً كحد أقصى. ويحذف الدليل إذا لم يكن فيه أى ملفات، وتتبع مواصفات المسار نفس تكوين أسماء المسارات في DOS، ويجب أن توضع مواصفات الملف بين علامتي تنصيص مزبوجتين إذا كان حرفياً.

وتستخدم عبارة RMDIR في حذف أدلة موجودة أثناء تنفيذ البرنامج. وفيما يلي بعض

الأمثلة:

```
RMDIR "C:\BASIC\TEMP"
RMDIR "\SALES\OLD.DAT"
RMDIR "\SCRATCH"
```

التطبيقات

عبارات FILES و CHDIR و MKDIR و RMDIR مفيدة جداً في الاتصال بالقرص والتعامل معه أثناء تنفيذ البرنامج. والاستخدامات محدودة بالتطبيق المبرمج فقط.

عملية تقليدية

توضح العملية التالية استخدام عبارات FILES و CHDIR و MKDIR و RMDIR. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :

```
File Edit View Search Run Debug Calls - File Help
<Untitled>
This program demonstrates the FILES, CHDIR, MKDIR, and RMDIR statements.
CLS
PRINT : PRINT
PRINT "First we create a directory so that we can change directories !"
PRINT "The directory will be called A:\TEMPQB.ILL."
PRINT "The directory will be created in drive A:,"; so; make; sure; you; ""
PRINT "have a formatted diskette in that drive with at least one file on it."
MKDIR "A:\TEMPQB.ILL"
PRINT
PRINT "Now we change to the root directory and list that directory."
INPUT "Press Enter to continue": CS
CHDIR "A:\\"
FILES "A:\\"

INPUT "Press Enter to continue": CS
INPUT "Press Enter to continue": CS
PRINT : PRINT "Now we change to the new directory, and list that directory."
INPUT "Press Enter to continue": CS
CHDIR "A:\TEMPQB.ILL"
FILES "A:\TEMPQB.ILL\*,*"

INPUT "Press Enter to continue": CS
PRINT
PRINT "Now we change back to the root and delete the directory we created."
PRINT "And we will show the directory to prove it !"
INPUT "Press Enter to continue": CS
CHDIR "A:\\"
RMDIR "A:\TEMPQB.ILL"
FILES "A:\\"

Immediate
Main: <Untitled> Context: Program not running 088 14:011
```

٢ - نفذ البرنامج واتبع تعليماته. لاحظ تأثير عبارات FILES و CHDIR و MKDIR و RMDIR على حالة القرص.

First we create a directory so that we can change directories !
The directory will be called A:TEMPQB.ILL.
The directory will be created in drive A:, so make sure you
have a formatted diskette in that drive with at least 1 file on it.

Now we change to the root directory and list that directory.

Press Enter to continue?

```
A:\
README .TXT      TEMPQB .ILL<DIR>
346112 Bytes free
Press Enter to continue?
```

Now we change to the new directory, and list that directory.

Press Enter to continue?

```
A:\TEMPQB.ILL
. <DIR>          .. <DIR>
346112 Bytes free
Press Enter to continue?
```

Now we change back to the root and delete the directory we created.
And we will show the directory to prove it !

Press Enter to continue?

The directory will be created in drive A:, so make sure you
have a formatted diskette in that drive with at least one file on it.

Now we change to the root directory and list that directory.

Press Enter to continue?

```
A:\
README .TXT      TEMPQB .ILL<DIR>
346112 Bytes free
Press Enter to continue?
```

Now we change to the new directory, and list that directory.

Press Enter to continue?

```
A:\TEMPQB.ILL
. <DIR>          .. <DIR>
346112 Bytes free
Press Enter to continue?
```

Now we change back to the root and delete the directory we created.
And we will show the directory to prove it !

Press Enter to continue?

```
A:\
README .TXT
347136 Bytes free
```

Press any key to continue

٣ - اضغط على أى مفتاح للعودة إلى البرنامج، من قائمة File اختر New واكتب N لإخلاء الشاشة.

٤ - انتقل إلى الدرس التسعين للاستمرار فى تسلسل التعلم.

الدرس الثامن والأربعون

دالة FIX

الوصف

تعيد دالة FIX الجزء الصحيح بعد الفاء الكسر العشري من التعبير العددي. وتكوينها هو كما يلي :

FIX(numeric expression)

الوصف	الجزء
كلمة بيسك سريع محجوزة العدد أو القيمة التي يلغى كسرهما العشري	FIX numeric expression

التطبيقات

تستخدم دالة FIX عندما تكون هناك حاجة إلى حذف الكسر العشري من عدد له دقة فردية أو دقة مزدوجة. مثال ذلك :

PRINT FIX(12.33)

المخرجات هي : 12

PRINT FIX(228.211)

المخرجات هي : 228

PRINT FIX(-811)

المخرجات هي : -811

PRINT FIX(-9009.0001)

المخرجات هي : -9009

عملية تقليدية

توضح العملية التالية دالة FIX. ابدأ بتحميل بيك السريع.

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls Help
<Untitled>
CLS
PRINT "FIX(-123.85) returns ": FIX(-123.85)
PRINT "INT(-123.85) returns ": INT(-123.85)
PRINT "FIX(123.85) returns ": FIX(123.85)
PRINT "INT(123.85) returns ": INT(123.85)

Immediate

Name: <Untitled> Context: Program not running 0000:0047

```

٢ - نفذ البرنامج، لاحظ مخرجات البرنامج واستخدام دالة FIX.

```

FIX(-123.85) returns -123
INT(-123.85) returns -124
FIX(123.85) returns 123
INT(123.85) returns 123

Press any key to continue

```

٣ - اضغط على أي مفتاح للعودة إلى البرنامج، اضغط على Alt-F واضغط على مفتاح الادخال واكتب N لإخلاء الشاشة.

٤ - انتقل إلى الدرس السادس والستين للاستمرار في تسلسل التعلم.

الدرس التاسع والأربعون

عبارة FOR.. NEXT

الوصف

العبارات التي تسمح لك بالتحكم في مسار البرنامج تسمى تكوينات تحكم-control structures. وعبارة FOR.. NEXT في البيسك هي مثل تكوين التحكم هذا. تنفذ عبارات البرنامج التي تقع داخل حدود دورة FOR.. NEXT عدداً معيناً من المرات. وتكوينها هو كما يلي:

```
FOR counter = numeric expression TO
    numeric expression [STEP numeric expression]
    program statements
NEXT counter
```

وأجزاء التكوين هي كما يلي :

الوصف	الجزء
كلمة بييسك محجوزة.	FOR
متغير عددي، لا يمكن أن يكون عنصراً من عناصر سجل أو منظومة في سجل.	counter
تعبير عددي صحيح، يمكن أن تكون القيمة عدداً مزيج الدقة، وتفضل القيم العددية الصحيحة لتحقيق السرعة.	numeric expression
كلمة بييسك محجوزة.	TO
جزء اختياري للتحكم في خطوة زيادة العداد counter	STEP
إذا كانت هناك حاجة إلى ذلك، وهي كلمة بييسك محجوزة.	
كلمة بييسك محجوزة تحدد نهاية العبارات التي تقع داخل دورة FOR.. NEXT.	NEXT

التداخل هو طريقة كتابة لسورات FOR.. NEXT بحيث إنه توجد دورات FOR.. NEXT أخرى داخل تكوين FOR.. NEXT. وعندما يحدث تداخل لعبارات FOR.. NEXT فيجب أن يوجد متغير عداد فردي لكل عبارة من هذه العبارات المتداخلة. وفيما يلي مثال يوضح بعض الطرق المختلفة لتكوين دورة FOR.. NEXT.

```

FOR Cnt1 = 1 TO 10
  ...
  FOR Cnt2 = 1 TO 10 STEP 2
    program statements
  ...
  FOR Cnt3 = 10 TO 1 STEP -1
    ...
    FOR Cnt4 = 1 TO LEN(Str00$)
      ...
    NEXT Cnt4
  NEXT Cnt3
NEXT Cnt2
NEXT Cnt1
(or)
NEXT Cnt4,Cnt3,Cnt2,Cnt1

```

لاحظ الأربعة أسطر لعبارات NEXT والتي يمكن استبدالها بعبارَة NEXT واحدة. لاحظ كذلك ترتيب العدادات في عبارة NEXT واحدة.

التطبيقات

عبارة FOR.. NEXT مفيدة جداً عندما يتطلب البرنامج تكرار تنفيذ مجموعة من التعليمات عدداً معيناً من المرات. وإمكانية تداخل عبارات FOR.. NEXT في بعضها تعتبر ميزة، وعادة ما تستخدم قيمة العداد counter داخل مجموعة FOR.. NEXT في الحسابات، وفيما يلي مثال لذلك.

```

k = 2
FOR i = 1 TO 10
  PRINT i, k*i
NEXT i

```

1	2
2	4
3	6
4	8
5	10
6	12
7	14
8	16
9	18
10	20

Press any key to continue

عملية تقليدية

توضح العملية التالية استخدام دورة FOR.. NEXT في برنامج. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls F1-Help
<Untitled>
CLS
INPUT "Enter rate per hour ", Rate
INPUT "Enter number of hours allowed per week ", Hours%
INPUT "Enter number of weeks worked ", Weeks%
PRINT : PRINT STRING$(40, "-")
PRINT "Week": SPACES(18): "Cumulative Wages": PRINT STRING$(40, "-")
FOR Cnt = 1 TO Weeks%
    PRINT Cnt: TAB(14): ((Rate * Hours%) * Cnt)
NEXT

```

Immediate

Home <Untitled> Console: Program not running 10/10/2011

٢ - اضغط على Shift-F5 لتنفيذ البرنامج. اكتب القيم 10.635 و 40 و 12 و لاحظ المخرجات.

```

Enter rate per hour 10.635
Enter number of hours allowed per week 40
Enter number of weeks worked 12

```

Week	Cumulative Wages
1	425.4
2	850.8
3	1276.2
4	1701.6
5	2127
6	2552.4
7	2977.8
8	3403.2
9	3828.6
10	4254
11	4679.4
12	5104.8

Press any key to continue

- ٣ - اضغط على أى مفتاح للعودة إلى قائمة البرنامج.
- ٤ - من قائمة File اختر Save. اكتب FORNEXT.BAS كاسم للملف ثم حدد صيغة الملف على أنها نصية واحفظ البرنامج. اضغط على Alt-F ثم اضغط على N لإخلاء الشاشة.
- ٥ - انتقل إلى الدرس الثامن والخمسين للاستمرار فى تسلسل التعلم.

الدرس الخمسون

دالة FRE

الوصف

تعطى دالة FRE كمية الذاكرة المتاحة فى مناطق مختلفة أثناء وقت التنفيذ، وتكوينها هو كما يلى:

FRE(number)
FRE(string)

نعيد دالة FRE، مع عدد كمؤشر، الذاكرة المتاحة فى المناطق التالية وذلك طبقاً لقيمة العدد.

العدد	القيمة
- 1	حجم أكبر منظومة ليست سلسلة يمكن أن يعد لها أبعاد
- 2	الكمية المتاحة لمكان الرصة.
other	حجم المجموعة التالية المتاحة لمخزن سلسلة.

والقيم التى تعود تكون كلها بالبايت.

وعندما يكون المؤشر سلسلة أو تعبير سلسلة فتعطى دالة FRE كمية التخزين المتاحة للسلسلة. كما أنها تضغط كذلك المخزن المتاح للسلسلة فى مجموعة واحدة.

التطبيقات

يمكن أن تعزز بحكمة دالة FRE تنفيذ البرنامج بطرق عديدة. وفيما يلى منطقتان لعمل ذلك: التاكيد من وجود ذاكرة كافية متاحة للمنظومات الديناميكية وانتاج تجاور فى توزيع مواقع السلسلة. وفيما يلى بعض أمثلة على دالة FRE.

```
DIM NewI(Max.Min)
RM = FRE(-1)
Max = Max + 2: Min = Min + 2
REDIM NewI(Max.Min)
```

يوضح المثال السابق كيف يمكن استخدام دالة FRE في تحديد ما إذا كانت هناك ذاكرة كافية متاحة لإعادة تحديد أبعاد المنظومة New1 أم لا.

```
AvMem = FRE("Junk")
```

يستخدم هذا المثال دالة FRE مع سلسلة ويتسبب في ضغط مخزن السلسلة المتاح في مجموعة واحدة، ويساعد ذلك في تحسين أداء البرنامج.

عملية تقليدية

تقوم في هذه العملية بتعديل البرنامج المقدم في الدرس المائة والسابع والثلاثين للتعامل مع دالة FRE. ابدأ بتحميل بيك السري.

١ - اختر فتح ملف Open وحمل البرنامج STA_DYN.BAS.

٢ - عدل الملاحظات وعبارات PRINT كما هو مبين لاستخدام مؤشرات مختلفة مع دالة FRE.

```

File Edit View Search Run Debug Calls Help
STA_DYN.BAS
' This program demonstrates the FRE function.
' The program declares arrays and deallocates them.
' The memory available before and after is displayed on the screen.

REM $STATIC
DIM U(100, 100)
REM $DYNAMIC
DIM Q(10000)
CLS
PRINT "Amount of free memory ";
PRINT FRE(-1); "/"; FRE(-2); "/"; FRE(10); "/"; FRE("Junk")

REDIM Q(1000)
PRINT "After redimensioning Q ";
PRINT FRE(-1); "/"; FRE(-2); "/"; FRE(10); "/"; FRE("Junk")

ERASE Q
PRINT "After erasing Q ";
PRINT FRE(-1); "/"; FRE(-2); "/"; FRE(10); "/"; FRE("Junk")

Main: STA_DYN.BAS   Compiled: Programmed: Running

```

٣ - نفذ البرنامج ولاحظ المخرجات. لاحظ أن الكل، باستثناء FRE (-1)، يعيد نفس القيمة في هذا البرنامج. وتكون المخرجات على النحو التالي :

```
Amount of free memory 262384 / 1162 / 49888 / 49888  
After redimensioning Q 298384 / 1162 / 49888 / 49888  
After erasing Q 302488 / 1162 / 49888 / 49888
```

Press any key to continue

٤ - ارجع إلى البرنامج مع اخلاء الشاشة دون أن تحفظ هذا البرنامج.

٥ - انتقل إلى الدرس المائة والثالث والعشرين للاستمرار في تسلسل التعلم.

الدرس الحادى والخمسون

دالة FREEFILE

الوصف

تعطى دالة FREEFILE رقم الملف التالى المتاح فى بيسك السريع وتكوينها هو كمايلى:

FREEFILE

والرقم الذى تعيده هو رقم ملف صحيح ولم يستخدم فى بيسك السريع. ويستخدم هذا الرقم فى تحديد رقم الملف فى عبارة OPEN.

التطبيقات

دالة FREEFILE مفيدة فى انتاج أرقام ملفات صحيحة أثناء تنفيذ البرنامج. ويمكن بواسطة دالة FREEFILE أن تنتج البرامج الفرعية والأجزاء الفرعية ملفات باستخدام أرقام الملفات التى تنتج عند وقت التنفيذ وفيما يلى أمثلة لدالة FREEFILE.

مثال ١

```
Input "Enter file name": FileName$
NextFile = FREEFILE
OPEN FileName$ FOR RANDOM AS NextFile
```

مثال ٢

```
SUB LoadFile(FileName AS STRING)
NextFN = FREEFILE
OPEN FileName FOR INPUT AS NextFN
END SUB
```

عملية تقليدية

تستخدم هذه العملية البرنامج الموجود فى الدرس الرابع والتسعين فى توضيح استخدام دالة FREEFILE. ابدأ بتحميل بيسك السريع.

١ - اختر Open من قائمة File وحمل البرنامج OPEN.BAS.

٢ - عدل عبارة OPEN وعبارة PRINT وسطر التعليق ليتواءما مع القائمة التالية :


```

File Edit View Search Run Debug Calls
OPEN.BAS
'This program demonstrates the FREEFILE statement.

CLS
ON ERROR GOTO FileError
PRINT : PRINT "Demonstration of the FREEFILE statement ..."
FILES "???.*"
PRINT : INPUT "Enter filename to view: "; FileName$

IF FileName$ <> "" THEN
    NextFile = FREEFILE
    OPEN FileName$ FOR INPUT AS NextFile
    PRINT : PRINT "Listing of file "; FileName$

    DO WHILE NOT EOF(NextFile)
        LINE INPUT #NextFile, InLine$
        PRINT InLine$
    LOOP

END IF

END

FileError:
PRINT "Abnormal (?) program termination.", ERR, ERL

END

```

Immediate

Name: OPEN.BAS Context: Program not running 00000000

٢ - نفذ البرنامج. لاحظ استخدام دالة FREEFILE في البرنامج. لاحظ كذلك أن الشاشة تبين قائمة مختلفة من الملفات. اختر ملفاً سبق لك حفظه في أحد الدروس السابقة. ويجب أن يكون الملف ملف ASCII لكي ينفذ البرنامج بطريقة صحيحة. وفيما يلي عينة للتنفيذ.

```

Demonstration of the FREEFILE statement ...
C:\QB
      . <DIR>          .. <DIR> BC      .EXE      QB      .EXE
LIB  .EXE      QB      .HLP      QB      .LIB      QB      .QLB
QB   .PIF      QB      .BI       BOX      .BAS      QB      .OBJ
BOX  .EXE      EXE      .MAP      ASC      .BAS      ABS      .BAS
UP   .TXT      BBZ      .DAT      SUB      .BAS      QB      .INI
BBZ  .BAS      A       .BAS
STR  .BAS      A       .BAS
2791424 Bytes free

Enter filename to view: ? ASC.BAS

Listing of file ASC.BAS
Num$ = "1234": Num% = 0
FOR i = 1 TO LEN(Num$)
    Num% = Num% + ((ASC(MID$(Num$, i, 1)) - 48) * (10 ^ (LEN(Num$) - i)))
NEXT i
PRINT Num$, Num%

Press any key to continue

```

- ٤ - ارجع إلى البرنامج، اختر New واختر عدم حفظ البرنامج.
- ٥ - انتقل إلى الدرس السادس والأربعين للاستمرار في تسلسل التعلم.

الدرس الثانى والخمسون

عبارة FUNCTION

الوصف

توضح وتعرف عبارة FUNCTION دالة فى بيسك السريع، وتكوينها هو كما يلى :

```
FUNCTION name (parameter list) STATIC
    name = expression
END FUNCTION
```

جزء name هو اسم الدالة وهو اسم متغير لبيسك سريع صحيح ولايزيد طوله عن 40 خانة. ويمكن أن يشمل الاسم معرفاً للنوع لتحديد نوع النتيجة التى تعيدها الدالة، وجزء parameter list هو قائمة المؤشرات التى تتوقعها الدالة وتقبلها، وتتحدد محتويات القائمة بفواصل، وقائمة المؤشرات اختيارية حيث يمكن تعريف الدالة بدون أى مؤشرات، (تكوين قائمة المؤشرات موصوف فى المقطع التالى)، الكلمة المحجوزة STATIC تستخدم اختيارياً لتحديد ما إذا كانت المتغيرات داخل الدالة تحتفظ بنفس قيمها بين الاستدعاءات أم لا. وجزء expression المحدد لاسم الدالة هو القيمة المراد تحديدها، وتعود كنتيجة للدالة، تحدد عبارة END FUNCTION انتهاء جسم الدالة.

وقائمة المؤشرات لها التكوين التالى :

```
var1 AS type, var2 AS type...
```

ويحدد جزء A Stype نوع بيانات المتغيرات، وبديلاً لذلك عندما يكون المتغير عبارة عن منظومة فيوضع جزء من القائمة على هيئة قوسين فارغين كما يلى :

```
var1(), var2()...
```

لاحظ غياب مواصفة A Stype.

التطبيقات

عبارة FUNCTION هى طريقة أخرى لعزل قطع من الشفرة أجرى عليها اختبار وتنفذ نشاطاً معرفاً تعريفاً جيداً، كل المتغيرات فى FUNCTION تكون محلية ويوضع لها قيم ابتدائية

مساوية اصفاً أو فراغات وذلك قبل تنفيذ الدالة. وتقدم الكلمة الرئيسية الاختيارية STATIC آلية لحفظ قيم المتغيرات المحلية بين استدعاءات الدالة كما هي. وحيث إن الدالة تعيد نتيجة واسم الدالة هو اسم متغير بيسك سريع فيستخدم اسم الدالة في عبارات التحديد. ويمكن للدالة المعرفة بعبارَة FUNCTION أن تكون لها خاصية الاعادة الذاتية أى إنها تستدعى نفسها. وعندما تتسم الدالة بأنها لها خاصية الاعادة الذاتية فلا يكون استخدام الكلمة المحجوزة STATIC فكرة طيبة مع الدالة. وفيما يلي بعض أمثلة لعبارات FUNCTION.

مثال ١

```
FUNCTION ConstStr$(Ch$ AS STRING)
    ..
END FUNCTION
```

مثال ٢

```
FUNCTION RandStr$(Range$ AS STRING)
    IF Range$ = "" THEN EXIT FUNCTION
    ..
END FUNCTION
```

مثال ٣

```
FUNCTION KeepCount(Tl()) STATIC
    ChCnt = ChCnt + 1
    ..
END FUNCTION
```

مثال ٤

```
FUNCTION Factorial(Num AS INTEGER)
    IF Num > 1 THEN Factorial = Num * Factorial(Num - 1)
    ..
END FUNCTION
```

يبين المثال الأخير كيف تتسم الدالة بخاصية الاعادة الذاتية.

عملية تقليدية

تستخدم هذه العملية البرنامج المقدم في الدرس السادس والثمانين. يقدم البرنامج توضيحاً جيداً لعبارَة FUNCTION. ابدأ بتحميل بيسك السريع.

١ - اختر Open وحمل البرنامج LRTRIM.BAS.

- ٢ - نفذ البرنامج ولاحظ استخدام عبارة FUNCTION في البرنامج.
- ٣ - اختر New مع اخلاء الشاشة دون أن تحفظ هذا البرنامج.
- ٤ - انتقل إلى الدرس المائة والسادس والثلاثين للاستمرار في تسلسل التعلم.

الدرس الثالث وخمسون

عبارتا GET و PUT

الوصف

عبارة GET : تقرأ عبارة GET بيانات من ملف في متغير أو ذاكرة احتياطية للاتصال العشوائي. وتكوينها هو كما يلي :

GET #filename, recordnum, variable

جزء filename هو رقم الملف المحدد للملف في عبارة OPEN. ويعطى جزء recordnum موضع بداية السجل لبدء قراءة البيانات. وفي ملفات الاتصال العشوائي يكون recordnum هو رقم السجل الفعلي. وفي الملفات الثنائية يكون رقم السجل هو موضع البايت الذي تبدأ القراءة عنده. وعندما يحذف هذا الجزء فتقرأ البيانات بدءاً بموقع انتهاء آخر سجل سبق قراءته. وأقصى رقم سجل هو 2,147,483,647. وجزء variable يتلقى البيانات المقروءة. وعند استخدام هذا الجزء فلا يكون ضرورياً استخدام نوال تحويل الحقل (CVS, CVD, CVL, CVI) ويجب عدم استخدام عبارة FIELD مع الملف المستخدم وذلك لأن عبارة FIELD تنظم البيانات بطريقة مختلفة عن عبارة GET. ومع وجود الملفات في الصيغة الثنائية فيمكن استخدام أى متغير وتقرأ عبارة GET عدد البايت كلها التي يسمح بها المتغير. وعندما تستخدم سلسلة متغيرة الطول كمقصد فتقرأ دالة GET عدداً من البايت يناظر ما يسمح به طول السلسلة الحالية. وجزئي re-cordnum و variable اختياريان.

عبارة PUT : تكتب عبارة PUT في ملف وذلك من متغير أو ذاكرة احتياطية للاتصال المباشر. وتكوينها هو كما يلي :

PUT #filename, recordnum, variable

والأجزاء شبيهة بأجزاء عبارة GET فيما عدى أن الغرض هو كتابة ملف بدلاً من القراءة من ملف. عند استخدام جزء variable فلا يكون ضرورياً استخدام نوال تحويل البيانات (MKD\$, MKS\$, MKI\$, MKL\$) ويجب ألا تستخدم عبارة FIELD مع الملف لنفس السبب المذكور مع عبارة GET.

التطبيقات

تسمح عبارات GET و PUT بطرق أكثر مرونة للاتصال بملفات اتصال عشوائية. فنسمح بالاتصال بملفات على المستوى الثنائي. ومع تقديمها المرونة فإنها تتطلب كذلك كما أكبر من المسئولية من قبل المترجم. وفيما يلي أمثلة عليها :

مثال ١

```
OPEN "Config.Dat" FOR RANDOM AS #1 LEN = 80
..
INPUT "Monochrome or Color "; ScrType$
..
PUT #1,,ScrType$
```

مثال ٢

```
TYPE NewDat
  F1 AS STRING * 20
  F2 AS STRING * 20
END TYPE
DIM DatRec AS NewDat
OPEN "New.Lst" FOR RANDOM AS #4 LEN = 40
FIELD #4 20 AS Field1, 20 AS Field2
..
GET #4
..
LINE INPUT #4, DatRec
..
```

يوضح مثال ٢ كيفية استخدام عبارة FIELD مع عبارة GET. لاحظ بصفة خاصة أن عبارة GET تستخدم بدون متغير. وتستخدم عبارة LINE INPUT في نقل بيانات من الذاكرة الاحتياطية لملف عشوائي إلى المتغير DatRec. كما يمكن استخدام عبارة INPUT كذلك. عندما تستخدم عبارات GET و PUT في الاتصالات مع سجلات ثابتة الطول فتنظر العبارات بلا نهاية لرموز رقم السجل. استخدم هذه السمة بحذر. وتكتب عبارات GET و PUT كما يلي عند ترك جزء أو أكثر من جزء بدون استخدام :

مثال ١

```
GET #1,,InputStr$ 'the recordnum is omitted
PUT #1,,InputStr$
```

مثال ٢

```
GET #2,2          'the variable is omitted
PUT #2,2
```

مثال ٢

```
GET #1          'the recordnum and variable are omitted
PUT #1
```

عملية تقليدية

توضح هذه العملية استخدام عبارات GET و PUT. ابدأ بتحميل بيك السريع.

١ - اكتب البرنامج التالي :

```

FILE  EDIT  View  Search  Run  Debug  Calls  F1-Help
<Untitled>
' This program demonstrates the use of GET and PUT statements. The program
' creates a file using the PUT statement and reads it back using the GET
' statement.
TYPE CustType
    CustName AS STRING * 25
    CustNum AS INTEGER
    CustType AS STRING * 2
END TYPE

DIM Customer AS CustType
OPEN "Cust.Fil" FOR RANDOM AS #2
RecCnt = 1
CLS
DO WHILE UCASE$(Choice$) <> "Y"
    INPUT "Enter Customer name: "; Customer.CustName
    INPUT "Enter Customer number: "; Customer.CustNum
    INPUT "Enter Customer type (A,B,C): "; Customer.CustType
    INPUT "Done ? (Y/N): "; Choice$
    PUT #2, RecCnt, Customer
    RecCnt = RecCnt + 1
LOOP
CLOSE #2

PRINT "Press any key to continue ..": INPUT ts
OPEN "Cust.Fil" FOR RANDOM AS #2
RecCnt = 1
GET #2, 1, Customer

DO WHILE NOT EOF(2)
    PRINT RecCnt; Customer.CustName; Customer.CustNum; Customer.CustType
    RecCnt = RecCnt + 1
    GET #2, RecCnt, Customer
LOOP

----- Immediate -----
Name: <Untitled>  Location: Program not running  000.00.000

```


٢ - نفذ البرنامج، اكتب البيانات التالية واضغط على مفتاح الادخال بعد ادخال كل مدخل. بعد آخر ملقن اضغط على أى مفتاح. لاحظ استخدام عبارات GET و PUT فى البرنامج.

```

Enter Customer name: ? Microsoft Inc.
Enter Customer number: ? 333
Enter Customer type (A,B,C): ? A
Done ? (Y/N): ? n
Enter Customer name: ? More Money Corp.
Enter Customer number: ? 1222
Enter Customer type (A,B,C): ? C
Done ? (Y/N): ? n
Enter Customer name: ? Singapore Scentis Co.
Enter Customer number: ? 999
Enter Customer type (A,B,C): ? B
Done ? (Y/N): ? Y
Press any key to continue ..
?
1 Microsoft Inc.                333                A
2 More Money Corp.             1222               C
3 Singapore Scentis Co.        999                B

Press any key to continue
    
```

٣ - اضغط على أى مفتاح للعودة إلى البرنامج، اختر New من قائمة File واكتب N لإخلاء الشاشة.

٤ - انتقل إلى الدرس الثامن والثلاثين للاستمرار فى تسلسل التعلم.

الدرس الرابع والخمسون

عبارات GET GRAPHICS و PUT GRAPHICS

الوصف

تتعامل عبارات GET و PUT المذكورة في هذا الدرس مع صور الرسومات وتنسخ عبارة GET نسخاً من صور الرسومات من الشاشة أما عبارة PUT فتعيد صور الرسومات المخزنة بواسطة عبارة GET إلى الشاشة. وتكوين العبارتين هو كما يلي :

```
GET STEP (x1,y1)-STEP(x2,y2),array
PUT STEP(x,y),array,action
```

عبارة GET : جزء STEP اختياري ويحدد أن الاحداثيات نسبية إلى آخر نقطة يشار إليها. ولايستخدم هذا الجزء مرتين في نفس عبارة GET سواء كانت احداثيات البداية نسبية أو كانت احداثيات النهاية نسبية أو كان كل من الاحداثيات للبداية والنهاية مطلقة. تشير أجزاء (x1, y1) و (x2, y2) إلى احداثيات البداية والنهاية لمنطقة مستطيلة يراد نسخها. وجزء array هو الموقع الذي تنسخ فيه منطقة الشاشة ويكون من النوع الصحيح. ويتحدد حجم المنظومة باستخدام الصيغة التالية :

$$\text{Size} = 4 + \text{INT}(((x2 - x1 - 1) * (\text{bits per pixel per plane} + 7) / 8) * \text{planes} * (y2 - y1) + 1)$$

تعتمد قيم المستويات وعدد البت في كل نقطة رسم واحدة في المستوى الواحد على الشاشة. ويعطى الجدول التالي البت لكل نقطة رسم لكل مستوى والمستويات لحالات الشاشة المختلفة.

المستويات	بت لكل نقطة رسم لكل مستوى	الحالة
1	2	1
1	1	2
4	1	7

المستويات	بت لكل نقطة رسم لكل مستوى	الحالة
4	1	8
2 (64K لذاكرة EGA)	1	9
4 (أكبر من 64K لذاكرة EGA)		
2	1	10
1	1	11
4	1	12
1	8	13

عبارة PUT : ينفذ جزء STEP نفس الوظيفة في عبارة PUT مثلاً يفعل في حالة عبارة GET. ويعطى جزء (x,y) إحداثيات البداية المراد وضع الهدف عندها. جزء array هو منظومة تحتوى على الهدف المحفوظ بواسطة عبارة GET. وجزء action يحدد كيفية وضع الصورة على الشاشة فوق الأشياء الموجودة. والجراءات actions هي PSET و PRESET و AND و OR و XOR. ويصف الجول التالي تقاطع الأشياء المراد وضعها مع أشياء الشاشة لكل من أفعال الإجراءات المسموح بها.

الوصف	الفعل
ينقل الشيء نقطة بنقطة وتكون الألوان هي نفسها مثل الألوان المراد نقلها.	PSET
مثل PSET فيما عدى وضع سالب للصورة.	PRESET
تتسبب في عملية اضافة AND على الشاشة والشيء المراد وضعه. وتظل الألوان المتشابهة في الشيء والشاشة كما هي. أما الألوان غير المتشابهة فتظل غير متشابهة.	AND
تتسبب في عملية أو OR على الشاشة والشيء المراد وضعه.	OR
تتسبب في عملية XOR على الشاشة والشيء المراد وضعه. يسمح بوضع الشيء المراد وضعه بدون حذف الصورة الموجودة.	XOR

تتسبب مؤثرات بوليان AND و OR و XOR في معالجة عبارة GET بنفس الطريقة التي تعالج بها المؤثرات المنطقية الأعداد.

التطبيقات

عبارات GET و PUT هي أكثر نفعاً في الرسوم المتحركة فيمكن استخدامها كذلك في أساليب عمل نوافذ الظهور pop-up في بيئة أعداد الرسومات. وفيما يلي بعض أمثلة لعبارات GET و PUT.

```
SCREEN 1
'$DYNAMIC
ArrSize = 1024
REDIM ObjArr(ArrSize)
GET (U1X,U1Y)-(Lrx,Lry),ObjArr
PUT (U1x,U1y),ObjArr,PSET
FUNCTION ASize(U1x,U1y,Lrx,Lry)
    ASize = 4+INT(((Lrx-U1x+1)*1+7)/8)*1*(Lry-U1y+1)
END FUNCTION
```

توضح الدالة FUNCTION ASize دالة حساب حجم حالة الشاشة المعطاة 2.

عملية تقليدية

هذه العملية هي توضيح بسيط لعبارات GET و PUT المستخدمة مع الرسومات. اكمل فقط إذا كانت لديك امكانيات رسومات ملونة تدعم ذلك. ابدأ بتحميل بيسك السريع.

١ - حمل البرنامج DRAW.BAS وعدل البرنامج كما هو مبين في القائمة التالية.

```

File Edit View Search Run Debug Calls F1=Help
DRAW.BAS
' This program demonstrates the use of the GET and PUT statements.

' Declare arrays as dynamic
$DYNAMIC

' Draw the face on the screen
SCREEN 1
DRAW "c1 u5 r5 d5 15 "
DRAW "bm+28,0 u5 r5 d5 15 "
DRAW "bm-8,0 d10"
DRAW "bm-5,5 r10"
DRAW "bm+15,0 u30 135"
xp = POINT(0): yp = POINT(1)
DRAW "d30 f10 r15 e10"
DRAW "bm=" + VARPTR(xp) + ",=" + VARPTR(yp)

FOR Cnt = 1 TO 12
    DRAW "c2 ne5 "
    DRAW "bm+3,0"
NEXT

' Calculate array size
asize = 4 + INT((65 * 2 + 7) / 8) * 1 * 65
asize = asize / 2
' Redimension the array, and create a second array to hold a piece of the
' blank screen
REDIM ObjArr(asize) AS INTEGER, ObjBlank(asize) AS INTEGER

' Get the object
GET (143, 70)-(158, 125), ObjArr
' Get a piece of the blank screen
GET (1, 1)-(65, 65), ObjBlank

' Put the blank piece on the object
PUT (143, 70), ObjBlank, AND
' Put the object at the top left corner of the screen
PUT (1, 1), ObjArr%

Immediate

Main: DRAW.BAS Context: Program not running 000004:001

```

٢ - نفذ البرنامج ولاحظ استخدام عبارات GET و PUT المستخدمة مع الرسومات في البرنامج.

٣ - ارجع إلى البرنامج واحفظ البرنامج كملف نصي تحت الاسم GPGGRAPH.BAS مع اخلاء الشاشة.

٤ - انتقل إلى الدرس الرابع والثمانين للاستمرار في تسلسل التعلم.

الدرس الخامس والخمسون

تكوين GOSUB.. RETURN

الوصف

يستخدم التكوين GOSUB.. RETURN فى النداء على برنامج فرعى والعودة منه. ولايشمل ذلك برامج فرعية سبق توضيحها باستخدام عبارات SUB أو FUNCTION. وتكوينه هو كما يلى :

GOSUB line number or line label
RETURN line number or line label

الوصف	الجزء
كلمة من كلمات بيسك المحجوزة تعمل تفريعاً إلى برنامج فرعى بدءاً برقم سطر معين أو باسم سطر معين. رقم السطر أو اسم السطر الذى يتم التفريع إليه.	GOSUB line number or line label
كلمة من كلمات بيسك المحجوزة التى تعيد التحكم إلى العبارة الموجودة عند رقم سطر معين أو اسم سطر معين على مستوى الجزء (البرنامج) فقط. فإذا لم يكن محدداً رقماً أو اسماً للسطر فيعود البرنامج إلى العبارة التى تلى عبارة .GOSUB	RETURN

يمكن استدعاء البرامج الفرعية أى عدد من المرات، كما يمكن استدعاؤها كذلك من برامج فرعية أخرى وداخل نفسها. والقيد الوحيد على عدد البرامج الفرعية التى يمكن أن تتداخل مع بعضها البعض هو مكان الرصة المتاح.

التطبيقات

البرامج الفرعية عبارة عن طريقة جيدة لعزل قطع من الشفرة يتكرر استخدامها فى أجزاء مختلفة من البرامج. ويساعد مثل هذا العزل على إعداد برمجة فعالة وعلى تسهيل قراءة البرنامج

وتسهيل صيانته كذلك. وتكوين GOSUB.. RETURN هو طريقة فعالة لاستدعاء برامج فرعية غير موضحة باستخدام عبارات SUB أو FUNCTION. وفيما يلي بعض الأمثلة :

مثال ١

```
GOSUB 200
..
200 PRINT "The result is " Qrt%
..
RETURN
```

هذا مثال لاستدعاء بسيط لبرنامج فرعي باستخدام GOSUB.. RETURN.

مثال ٢

```
GOSUB 1500: GOSUB 2000: GOSUB CheckList
```

هذا مثال لكيفية كتابة شفرة أكثر من عبارة GOSUB واحدة في نفس السطر. وحيث إن عبارة RETURN تعود إلى عبارة البرنامج التي تلي GOSUB فتستدعي البرامج الفرعية واحداً تلو الآخر.

مثال ٣

```
GOSUB MoveData
..
ContMove:
..
MoveData:
..
RETURN ContMove
```

هذا مثال لكيفية عودة عبارة RETURN إلى عبارة من عبارات البرنامج غير العبارة التي تلي GOSUB مباشرة. وفي هذه الحالة تمرر RETURN التحكم إلى الاسم ContMove.

مثال ٤

```
GOSUB Truncate
..
Psl:
..
Truncate:
..
GOSUB StripBlanks: GOSUB Tokenize
..
Tsl:
..
RETURN Psl
StripBlanks:
..
RETURN Tsl
Tokenize:
..
RETURN
```

يبين هذا المثال كيفية تداخل البرامج الفرعية في بعضها البعض فالبرنامج الفرعي الذي يبدأ عند الاسم Truncate يستدعي البرنامجين الفرعيين الموجودين عند الاسمين StripBlanks

و Tokenize. من الممكن عمل شفرة لعبارة GOSUB على مستوى StripBlanks و Token- ize كذلك. لاحظ أن الاسم المستخدم في عبارة RETURN الموجودة في Truncate يشير إلى Ps1 والاسم المستخدم في عبارة RETURN الموجودة في StripBlanks يشير إلى Ts1. وهذا بسبب أنه ليس من الممكن العودة إلى مدى أكبر من المدى الذي استدعى البرنامج الفرعي. وهذا يعنى أن البرنامج لا يستطيع العودة إلى Ps1 من StripBlanks ويترك Truncate حيث إنه لم يستدع من هذا المستوى.

مثال

```
GOSUB Factorial
...
Factorial:
  IF (f1 <= f2) THEN
    f1 := f1 - 1
    GOSUB Factorial
  ELSE RETURN
END IF
```

هذا مثال لكيفية استدعاء برنامج فرعي لنفسه. فالبرنامج الفرعي الموجود عند الاسم -Factorial يستدعي نفسه حيث يتم حساب المضروب. ويعرف هذا بالإعادة الذاتية recursion. هذا النوع من الشفرة يجب أن يستخدم بحذر شديد لتجنب إثارة مكان الرصة وحدث انهيار للبرنامج.

عملية تقليدية

هذا المثال يوضح استخدام تكوين RETURN.. GOSUB. ابدأ بتحميل بيك السريع.

١ - اكتب البرنامج التالي :


```

File Edit View Search Run Debug Calls F1-Help
<Untitled>

CLS
Sel1$ = "Selection 1": Sel2$ = "Selection 2": Sel3$ = "Selection 3"
Sel4$ = "Quit": Sel5$ = "Enter the number, or Q to quit"
LOCATE 2, 30: PRINT "MENU SELECTION"
LOCATE 4, 30: PRINT Sel1$
LOCATE 5, 30: PRINT Sel2$
LOCATE 6, 30: PRINT Sel3$
LOCATE 7, 30: PRINT Sel4$
LOCATE 8, 30: PRINT Sel5$
Y = 4: X = 30
GOSUB GetKey

DO WHILE ch$ < > "Q"
    IF ch$ = "1" THEN
        GOSUB Select1
    ELSEIF ch$ = "2" THEN GOSUB Select2
    ELSEIF ch$ = "3" THEN GOSUB Select3
    END IF
    GOSUB GetKey
LOOP

END

GetKey:
    ch$ = ""
    DO WHILE ch$ = ""
        ch$ = INKEY$
    LOOP
    ch$ = UCASE$(ch$)
    RETURN

Select1:
    LOCATE 10, 30: PRINT "Selection One "
    RETURN

Select2:
    LOCATE 10, 30: PRINT "Selection Two "
    RETURN

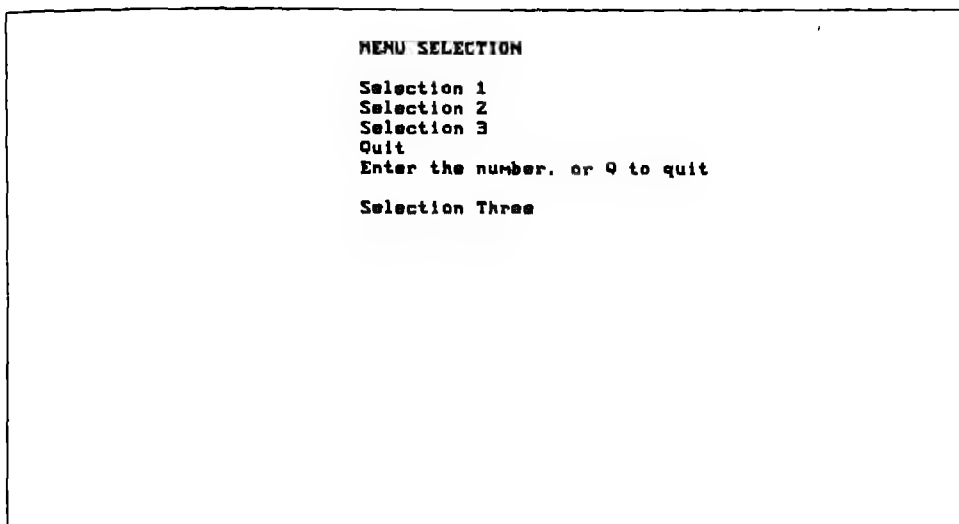
Select3:
    LOCATE 10, 30: PRINT "Selection Three"
    RETURN

Immediate

Main: <Untitled> Context: Program not running 0000000000000000

```

٢ - اضغط على Shift-F5 للتنفيذ. لاحظ كيفية عمل RETURN.. GOSUB في البرنامج. اكتب Q لإنهاء البرنامج.



- ٣ - اضغط على أى مفتاح للعودة إلى قائمة البرنامج.
- ٤ - اختر Save من قائمة File واكتب GOSUB.BAS كاسم للملف. حدد الملف من النوع النصي واحفظ هذا البرنامج.
- ٥ - اختر New من قائمة File لإخلاء الشاشة.
- ٦ - انتقل إلى الدرس السادس والثلاثين للاستمرار فى تسلسل التعلم.

الدرس السادس والخمسون

عبارة GOTO

الوصف

عبارة GOTO هي تفريع غير شرطى، وهذا يعنى أن البرنامج يستمر على أى حال من الأحوال مع السطر أو الاسم الذى تشير إليه عبارة GOTO. ويمكن أن يكون التفريع غير الشرطى على نفس مستوى عبارة GOTO فقط، ولا يمكنك أن تستخدم GOTO فى الدخول فى برنامج فرعى أو دالة أو الخروج منها يكون معرفاً (أو معرفة) بواسطة SUB أو FUNCTION أو DEF FN. وتكوينها هو كما يلى :

GOTO line number|line label

الوصف	الجزء
كلمة من كلمات بيسك السريع المحجوزة. رقم صحيح لسطر، يتراوح من 0 إلى 65,529، أو اسم صحيح لسطر يبدأ بحرف. وهذا هو مقصد أمر التفريع.	GOTO linenum 1 line label

التطبيقات

الاستخدامات عديدة إلا أن الاستخدامات الكثيرة وغير المميزة لعبارة GOTO يمكن أن تقود إلى شفرة تشبه المكرونة الاسباكيلى (صعبة التصحيح وصعبة القراءة والفهم وصعبة الصيانة). ويوصى بشدة بالتحكم فى مسار البرنامج باستخدام العبارات IF.. THEN.. ELSE و DO.. LOOP و FOR.. NEXT و WHILE.. WEND و SELECT.. CASE وذلك بدلاً من استخدام عبارات GOTO.

```
GOTO ErrorRoutine
GOTO 233
GOTO Terminate
```

عملية تقليدية

يوضح هذا القسم استخدام عبارات GOTO. استمر على النحو التالى :

١ - إذا كنت تستمر من درس إلى آخر فعليك باخلاء منقح بيسك السريع وذلك باختيار New من قائمة File مع اختيار عدم حفظ البرنامج المحمل حالياً وإلا فابدأ بيسك السريع من البداية.

٢ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls F1-Help
<Untitled>
' This program demonstrates the GOTO statement.
CLS
INPUT "Enter a value ."; v
IF v < 10 THEN GOTO LessNTen
IF v < 20 AND v > 10 THEN GOTO LessNTwenty
IF v > 20 AND v < 30 THEN GOTO LessNThirty
PRINT "Value more than 30."; END
LessNTen: PRINT "Value is less than 10."; END
LessNTwenty: PRINT "Value is more than 10 and less than 20."; END
LessNThirty: PRINT "Value is more than 20 and less than 30."; END

Immediate

Name: <Untitled> Context: Program not running

```

٣ - اضغط على Shift-F5 لتنفيذ البرنامج.

```

Enter a value .7 1
Value is less than 10.

Press any key to continue

```

٤ - اضغط على أى مفتاح للعودة إلى قائمة البرنامج. اضغط على Alt-F واكتب F لحفظ هذا البرنامج.

٥ - اكتب GOTO واضغط على Tab واستخدم مفاتيح الأسهم لتحديد أن الملف نصي. اضغط على Tab وعلى مفتاح الإدخال لحفظ هذا البرنامج. تذكر أن بيسك السريع يضيف التوسع لاسم الملف BAS.

٦ - انتقل إلى الدرس السبعين للاستمرار فى تسلسل التعلم.

الدرس السابع والخمسون

دالة HEX\$

الوصف

تعيد دالة HEX\$ القيمة السادسة عشرية لتعبير عددي عشري، وتكوينها هو كما يلي :

HEX\$(numeric expression)

يقرب التعبير العددي إلى أقرب قيمة صحيحة، فإذا وقعت القيمة خارج المدى الصحيح (من 0 إلى 32767) فإنه يتحول إلى قيمة صحيحة طويلة في الصورة السادسة عشرية. والقيمة التي تعود تكون سلسلة ولا يمكن استخدامها على هذا في الحسابات.

التطبيقات

تستخدم دالة HEX\$ عندما تكون هناك حاجة إلى معرفة القيمة السادسة عشرية لتعبير عددي عشري، ومثل هذا التحويل مفيد للمبرمجين الذين يتعاملون مع مثل هذا التمثيل للبيانات، وفيما يلي بعض أمثلة لدالة HEX\$.

```
PRINT HEX$(4096)
H$ = HEX$(128*64)
T$ = HEX$(12/3)
T$ = HEX$(1.09*13)
```

عملية تقليدية

دالة HEX\$ موضحة في العملية التالية. ابدأ بتحميل بيסק السريع.

١ - اكتب البرنامج التالي :

```

File Edit View System Run Debug Calls F1-Help
<Untitled>
CLS
FOR A% = 33 TO 128
  AHex% = ASC(CHR$(A%))
  PRINT CHR$(A%); " "; HEX$(AHex%); " ";
NEXT
Immediate
Name: <Untitled> Context: Program not running 00004:011

```

٢ - نفذ البرنامج. لاحظ أن البرنامج يطبع رموزاً مقرونة من جدول ASCII (من 33 إلى 128) والمكافئ السداس عشري لقيمها ASCII.

```

! 21/ " 22/ # 23/ $ 24/ % 25/ & 26/ ' 27/ ( 28/ ) 29/ * 2A/ + 2B/ , 2C/ - 2D/ .
2E/ / 2F/ 0 30/ 1 31/ 2 32/ 3 33/ 4 34/ 5 35/ 6 36/ 7 37/ 8 38/ 9 39/ : 3A/ ; 3B/
/ < 3C/ = 3D/ > 3E/ ? 3F/ @ 40/ A 41/ B 42/ C 43/ D 44/ E 45/ F 46/ G 47/ H 48/
I 49/ J 4A/ K 4B/ L 4C/ M 4D/ N 4E/ O 4F/ P 50/ Q 51/ R 52/ S 53/ T 54/ U 55/ U
56/ V 57/ X 58/ Y 59/ Z 5A/ [ 5B/ \ 5C/ ] 5D/ ^ 5E/ _ 5F/ ` 60/ a 61/ b 62/ c 63/
/ d 64/ e 65/ f 66/ g 67/ h 68/ i 69/ j 6A/ k 6B/ l 6C/ m 6D/ n 6E/ o 6F/ p 70/
q 71/ r 72/ s 73/ t 74/ u 75/ v 76/ w 77/ x 78/ y 79/ z 7A/ { 7B/ | 7C/ } 7D/ ~
7E/ ^ 7F/ 80/
Press any key to continue

```

٣ - اضغط على أى مفتاح للعودة إلى البرنامج. اختر New من قائمة File واكتب N لإخلاء الشاشة.

٤ - انتقل إلى الدرس الحادى والتسعون للاستمرار فى تسلسل التعلم.

الدرس الثامن والخمسون

عبارة IF..THEN..ELSE

الوصف

تضع معظم البرامج شروطاً تحتاج لأن تتحقق قبل أن يمكن تنفيذ جزء معين من الشفرة. ويعرف هذا بأنه تفريع شرطي conditional branching. ويتحقق ذلك في بيسك السريع باستخدام عبارات IF..THEN..ELSE. وهناك طريقتان لكتابة عبارة IF..THEN..ELSE. الطريقة الأولى تستخدم تكوين سطر واحد له الشكل التالي :

IF boolean expression THEN statement ELSE statement

الوصف	الجزء
كلمة من كلمات بيسك السريع المحجوزة.	IF
تعبير تختبر نتيجته لأن تكون صحيح أو خطأ. تنفذ العبارات التي تأتي بعد THEN إذا كانت نتيجة التعبير صحيح أما إذا كانت النتيجة خطأ فتنفذ العبارات التي تلي ELSE.	boolean expression
كلمة من كلمات بيسك السريع المحجوزة.	THEN
عبارة من عبارات بيسك السريع.	statement
كلمة من كلمات بيسك السريع المحجوزة.	ELSE
عبارة من عبارات بيسك السريع.	statement

تستخدم الطريقة الثانية تكوين مجموعة بالشكل التالي :

```
IF boolean expression THEN
statement block 1
ELSEIF boolean expression THEN
statement block 2
ELSE
statement block 3
END IF
```


والأجزاء الجديدة فى الطريقة الثانية موصوفة على النحو التالى :

الجزء	الوصف
statement block1, 2, and 3 ELSE IF	عبارات برنامج ببسك السريع التى يراد تنفيذها عندما يتحقق الشرط. كلمة من كلمات ببسك السريع المحجوزة. تستخدم فى التداخل فى عبارة IF..THEN.
END IF	كلمة من كلمات ببسك السريع المحجوزة. وهى مطلوبة لانتهاء تكوين IF..THEN فى الطريقة الثانية.

فى الطريقة الأولى يمكن أن تحتوى عبارات ببسك التى تنفذ بعد THEN أو ELSE على تعليمات تفريعات اضافية مثل GOTO و GOSUB. فإذا ما أشار التفريع إلى رقم سطر فتكون GOTO اختيارية. وإذا ما أشار التفريع إلى اسم سطر فتكون الكلمات المحجوزة GOTO أو GOSUB لازمة مع الاسم.

وفى كل من الطريقتين يكون جزء ELSE من التكوين اختيارياً، فإذا لم يكن الجزء ELSE موجوداً فيستمر البرنامج فى التنفيذ من عند العبارة التالية.

التطبيقات

تكوين IF..THEN..ELSE عبارة عن وسيلة جيدة جداً فى أى برنامج يأخذ قرارات مبنية على شروط تحدث أثناء تنفيذ البرنامج. الاستخدام الحكيم لتكوينات التفريع المتاحة مع طريقة المجموعة فى ببسك السريع يجعل البرامج ممتعة فى كتابتها وقراءتها وصيانتها. وفيما يلى بعض الأمثلة :

الطريقة الأولى:

```
IF (Hours% > RegHrs%) THEN GOTO 800 ELSE 1800
IF (Hours% > RegHrs%) THEN OverTime ELSE RegTime
IF (Q = SValue) THEN Q=Q^2:SValue=SQR(SValue):PP=SValue+Q: ELSE 200
```

الطريقة الثانية:

```

IF (OutStanding! <= CreditLimit!) THEN
  GOSUB AccountsUpdate
  GOSUB LedgerEntry
  GOSUB JE
ELSE
  GOSUB OverDueAccts
  GOSUB PayableAging
END IF

IF Switch = 1 THEN
  GOSUB Step1: GOSUB Step23
  CALL Evaluate
ELSEIF Switch = 2 THEN
  GOSUB Step2: GOSUB Step24
  CALL Evaluate
ELSEIF Switch = 3 THEN
  GOSUB Step3: GOSUB Step25
END IF
END IF
END IF

```

عملية تقليدية

يوضح البرنامج التالي استخدام عبارة IF..THEN..ELSE. ابدأ بتحميل بيك السريع.

١ - اكتب البرنامج التالي :

```

CLS
PRINT "This is a guessing game."
PRINT "I have a number in mind. You try to guess what it is."
RANDOMIZE TIMER
Guess% = INT(10 - 1) * RND + 1
PRINT "You have three guesses..."
Cnt = 1
DO WHILE (Cnt <= 3)
  INPUT "Your guess ", InG%
  IF InG% > Guess% THEN PRINT "Too high.."
  IF InG% < Guess% THEN PRINT "Too low.."
  IF InG% = Guess% THEN PRINT "Bingo !": Cnt = 4
  Cnt = Cnt + 1
LOOP
PRINT "My number was "; Guess%; " and your number was "; InG%

```

٢ - اضغط على Shift-F5 لتنفيذ البرنامج وادخل ثلاثة تخمينات.

```
This is a guessing game.  
I have a number in mind. You try to guess what it is.  
You have three guesses...  
Your guess 4  
Too low..  
Your guess 9  
Too high..  
Your guess 5  
Too low..  
My number was 7 and your number was 5
```

Press any key to continue

٣ - اضغط على أى مفتاح للعودة إلى قائمة البرنامج.

٤ - من قائمة File اختر Save واكتب IFTHEN.BAS حدد شكل الملف أنه من النوع النصي واحفظ هذا البرنامج.

٥ - اختر New من قائمة File مع اخلاء الشاشة.

٦ - انتقل إلى الدرس المائة والسادس والعشرين للاستمرار فى تسلسل التعلم.

الدرس التاسع والخمسون

عبارة \$INCLUDE

الوصف

عبارة \$INCLUDE تسمى بشبيه الأمر مثل \$STATIC و \$DYNAMIC. ويسمح شبيه الأمر \$INCLUDE بتشغيل ملفات مصدر أخرى أثناء الترجمة. وتكوين العبارة هو كما يلي :

```
REM $INCLUDE: 'filename'
```

ومثل كل اشياء الأوامر يظهر أمر \$INCLUDE في عبارة REM. جزء filename هو الملف المراد تشغيله أثناء الترجمة ويمكن أن يشمل اسم مسار. وبعد انتهاء تشغيل الملف (القراءة والترجمة) تستمر الترجمة بالعبارة التي تلى أمر \$INCLUDE مباشرة. والقيود الموجودة على الملفات المشمولة هي كما يلي :

- يجب ألا تحتوى الملفات على عبارات SUB أو عبارات FUNCTION.
- يجب أن تحفظ الملفات التي سبق انتاجها باستخدام صيغ بيسك القديمة عن طريق خيار A أى كملف ASCII.

التطبيقات

يستخدم شبيه الأمر \$INCLUDE في احتواء ملف خارجي أثناء الترجمة مثل توضيحات ملفات SUB و FUNCTION. وفيما يلي أمثلة لشبيه الأمر \$INCLUDE :

```
REM $INCLUDE: 'Declare1.bi'
'$INCLUDE: 'Absolute.Bi'
```

عملية تقليدية

توضح هذه العملية شبيه الأمر \$INCLUDE عن طريق استخدامه في تحميل ملف توضيحات. إذا كانت لديك أى مشاكل مع أى من هذه الخطوات أرجع إلى الملحق B (اقطع والصق، استخدام صناديق الحوار، قائمة أوامر التنقيح). ابدأ بتحميل بيسك السريع.

١ - اختر Open من قائمة File وحمل البرنامج SUB.BAS.

٢ - انقل نقطة البداية إلى عبارة DECLARE واضغط على مفتاح السهم السفلى Shift- واضغط على Shift-Del. يحذف ذلك السطر مع وضعه في لوحة القص.

٣ - اختر Create من قائمة File واكتب اسم الملف DECLARE.BI وحدد نوع الملف بأنه In- clude واضغط على مفتاح الإدخال. اضغط على Shift-Ins والصق السطر الموجود في لوحة القص.

```
File Edit View Search Run Debug Calls F1=Help
DECLARE.BI
DECLARE SUB SortArray (A%( ))
----- Immediate -----
Main: DECLARE.BI Context: Program not running 00001:001
```

٤ - اضغط على F2 واختر SUB.BAS لتنقيحه واضف عبارة \$INCLUDE مع أى اسم مسار لازم. ارجع إلى القائمة التالية :

```
File Edit View Search Run Debug Calls F1=Help
SUB.BAS
'This program illustrates the use of the SUB statement. The program
'sorts an array in ascending order.
'$INCLUDE: 'Declare.BI'
Max = 15
DIM A%(Max)
FOR Cnt = 1 TO Max
    READ A%(Cnt)
NEXT
CALL SortArray(A%( ))
CLS : PRINT
PRINT "The sorted array:"
FOR Cnt = 1 TO Max
    PRINT A%(Cnt);
NEXT
DATA 12,23,789,90,545,22,1,87,43,53,52,333,24,67,88

File Edit View Search Run Debug Calls F1=Help
SUB.BAS:SortArray
SUB SortArray (A%( ))
FOR C1 = LBOUND(A%( )) TO UBOUND(A%( )) - 1
    FOR C2 = C1 + 1 TO UBOUND(A%( ))
        IF A%(C1) > A%(C2) THEN
            Temp = A%(C2)
            A%(C2) = A%(C1)
            A%(C1) = Temp
        END IF
    NEXT C2
NEXT C1
END SUB
----- Immediate -----
Main: DECLARE.BI Context: Program not running 00001:001
```

٥ - نفذ البرنامج ولاحظ تأثير عبارة \$INCLUDE.

٦ - اختر Save As من قائمة File واحفظ هذا البرنامج كملف نصي تحت اسم ملف IN-CLUE.BAS مع اخلاء الشاشة.

٧ - انتقل إلى الدرس الحادي والعشرين للاستمرار في تسلسل التعلم.

الدرس الستون

دالة INKEY\$

الوصف

تعيد دالة INKEY\$ سلسلة من بايت واحد أو اثنين طبقاً للمفتاح المضغوط عليه من لوحة المفاتيح. وتكوين دالة INKEY\$ هو كما يلي :

INKEY\$

إذا كان المفتاح المضغوط عليه هو أحد الرموز القابلة للطباعة فيعيد بايت واحد. أما إذا كان المفتاح المضغوط عليه خليطاً من مشاوير مفاتيح مثل Ctrl-X أو Alt-D فنعيد INKEY\$ سلسلة من اثنين بايت. وفي هذه الحالة تحتوى البايث الأولى على رمز الفراغ وتحتوى البايث الثانية على رمز الفحص المتسع. يحتوى الملحق B على قائمة برموز فحص لوحة المفاتيح.

ولاتصدر دالة INKEY\$ صدى للرموز التى تقرأ من لوحة المفاتيح فى الشاشة. ويمكن أن تستخدم INKEY\$ كذلك مع وحدات نمطية أخرى إلا أن لوحة المفاتيح هى الأكثر استخداماً. وتمر كل الرموز التى يتم ادخالها إلى البرنامج باستثناء ما يلي :

ينهى البرنامج	Ctrl-Break
يعيد بدء عمل النظام	Ctrl-Alt-Del
يوقف تنفيذ البرامج	Ctrl-Numlock
يطبع محتويات الشاشة على الطابع	PrtSc

فى البرامج القائمة بذاتها (.EXE) يقرأ تسلسل Ctrl-Break إذا لم يكن خيار d /محدداً أثناء وقت الترجمة.

التطبيقات

دالة INKEY\$ عبارة عن وسيلة مفيدة فى قراءة مفاتيح متسعة من لوحة المفاتيح. وتقدم معالجة سلسلة الرموز التى تعود من الدالة آلية قوية للتحكم فى تداخل البرنامج مع المستخدم.

وفيما يلي بعض الأمثلة لاستخدام INKEY\$.

```
DO
LOOP UNTIL (INKEY$ <> "")
Ch$ = INKEY$
Ch1$ = MID$(Ch$,1,1): Ch2$ = MID$(Ch$,2)
GOSUB ProcessKey
```

أول مثال يستخدم INKEY\$ فى الانتظار حتى يتم الضغط على أى مفتاح. ويستخدم المثال الثانى دالة INKEY\$ فى الحصول على مشوار مفتاح وتجزئة جزئى السلسلة والتفرع إلى برنامج فرعى سوف يجرى تشغيلاً على جزئى مشوار المفاتيح ويفسر مشوار المفتاح.

عملية تقليدية

هذه العملية توضح كيف يمكن استخدام دالة INKEY\$ فى تشغيل مشاوير المفاتيح. يبحث البرنامج بصفة خاصة عن المفاتيح المتسعة التى يتم ادخالها.

ابدأ بتحميل ببسك السريع.

١ - اكتب البرنامج التالى :

```
File Edit View Search Run Debug Calls FI Help
INKEY.BAS
' This program demonstrates the use of the INKEY$ function.
CLS
PRINT "This program demonstrates how to find out if a combination"
PRINT "of keystrokes has been entered."
PRINT "The Alt and D through F key combinations are tested."

DO
  DO
    Ch$ = INKEY$
    IF Ch$ <> "" THEN GOSUB InterpretKey
  LOOP UNTIL (Ch$ <> "")
LOOP UNTIL (UCASE$(Ch$) = "Q")
END

InterpretKey:
Ch1$ = MID$(Ch$, 1, 1): Ch2$ = MID$(Ch$, 2)
IF ASC(Ch1$) = 0 THEN
  IF ASC(Ch2$) = 32 THEN PRINT "Alt-D was pressed"
  IF ASC(Ch2$) = 18 THEN PRINT "Alt-E was pressed"
  IF ASC(Ch2$) = 33 THEN PRINT "Alt-F was pressed"
END IF
RETURN

3. Immediate
```


٢ - نفذ البرنامج، اضغط على Alt-D و Alt-E اكتب Q للخروج. لاحظ استخدام دالة IN-KEY\$ لاكتشاف مشاوير المفاتيح المتسعة. وفيما يلي عينة للتنفيذ :

```
This program demonstrates how to find out if a combination
of keystrokes has been entered.
The Alt and D through F key combinations are tested.
Alt-D was pressed
Alt-E was pressed
Alt-F was pressed
```

Press any key to continue

٣ - اضغط على أى مفتاح للعودة إلى البرنامج. اختر New من قائمة File واكتب N لإخلاء الشاشة.

٤ - انتقل إلى الدرس الرابع والستين للاستمرار فى تسلسل التعلم.

الدرس الواحد وستون

عبارات INP و OUT و WAIT

الوصف

عبارة INP : تقرأ عبارة INP بايت من بوابة الآلة. وتكوينها هو كما يلي :

INP(port)

حيث port هو رقم صحيح يحدد بوابة صحيحة لمداخلات ومخرجات الكمبيوتر.

عبارة OUT : ترسل بايت إلى بوابة من بوابات الآلة. وتكوينها هو كما يلي :

OUT port,byte

حيث port هو بوابة صحيحة لمداخلات ومخرجات الكمبيوتر وجزء byte هو قيمة بيانات

ترسل إلى هذه البوابة. وتقع قيمة byte في المدى من 0 إلى 255.

عبارة WAIT : توقف تنفيذ البرنامج اثناء انتظاره لتطوير نمط بت غير صفرية في احدى

بوابات الآلة. وتكوينها هو كما يلي :

t, and-op, xor-op

وجزاء port عبارة عن رقم صحيح يحدد بوابة صحيحة لمداخلات ومخرجات الكمبيوتر. جزء

and-op هو تعبير عددي صحيح مدموج مع البيانات من البوابة في عملية AND. وجزء xor-op

هو تعبير عددي صحيح يكون مدموجاً مع بيانات من البوابة الموجودة في عملية XOR. تستخدم

عمليات بوليان من اليمين إلى اليسار أى إن عملية XOR تؤدي أولاً تليها عملية AND. عندما

يحذف xor-op فيفترض أنه صفر.

التطبيقات

عبارات INP و OUT و WAIT تستخدم في التعامل مباشرة مع نظم مكونات الكمبيوتر

وعلى هذا فيجب أن تستخدم مع الحرص الشديد. وبسبب الاختلافات في أجهزة الكمبيوتر

الشخصية ونظم مكوناتها، حتى عبر الأجهزة المتوافقة تماماً، فإن المعاملة المباشرة لبوابات

مدخلات ومخرجات الكمبيوتر يجب تجنبها كلما كان ذلك ممكناً. وفيما يلي أمثلة لعبارات INP

و OUT و WAIT:

```

InByte = INP 32
OUT 32,OutByte
IF InByte = .. THEN WAIT 32,9

```

عملية تقليدية

تقع مناقشة بوابات المدخلات والمخرجات المختلفة ومعالجتها خارج مدى هذا الكتاب وعلى ذلك فلا يحتوى هذا القسم على مثال لتوضيح هذه الأساسيات.

انتقل إلى الدرس الخامس والتسعين للاستمرار فى تسلسل التعلم.

الدرس الثانى والستون

عبارة INPUT

الوصف

تستخدم عبارة INPUT فى قبول مدخلات من لوحة المفاتيح اثناء تنفيذ البرنامج. وهذه هى احدى الطرق الأكثر استخداماً فى البيسك فى ادخال قيم المتغيرات. وتكوينها هو كما يلى :

INPUT (:) Prompt string (:) (,) variable list

الوصف	الجزء
كلمة من كلمات بييسك المحجوزة.	INPUT
مؤشر اختياري يتسبب فى أن تظل نقطة البداية موجودة على نفس السطر بعد أن يضغط المستفيد على مفتاح الادخال.	(;)
سلسلة اختيارية تعرض كملقن للمدخلات. ويجب أن توضع السلسلة بين علامتى تنصيص مزدوجتين.	Prompt String
مؤشر اختياري يتسبب فى طباعة علامة استفهام فى نهاية سلسلة الملحق.	(;)
مؤشر اختياري يتسبب فى ضغط علامة الاستفهام.	(,)
قائمة متغيرات تقبل فيها المدخلات. ويجب أن تفصل المتغيرات عن بعضها البعض بواسطة فواصل فى عبارة INPUT وفى استجابة INPUT.	variable list

توقف عبارة INPUT البرنامج وتطبع علامة استفهام وتنتظر المدخلات من لوحة المفاتيح. ويحدد الضغط على مفتاح الادخال انتهاء المدخلات. فإذا ما كانت القيمة التى يتم ادخالها من نوع غير صحيح أو أنها أكبر من اللازم أو أقل من اللازم فيقدم المترجم رسالة الخطأ التالية :

Redo from start

فعلى سبيل المثال استجابة الحرف "A" لطلب قيمة عددية ينتج عنه رسالة الخطأ. ويعطى الجدول التالى أمثلة أخرى :

Program line	User response	Valid
INPUT "Enter radius ", R%	12	Yes
	Ac	No
INPUT "Enter List", A\$, B\$	Hello,Dolly	Yes
	Hello	No
	Dolly	No
INPUT A\$,B\$,C\$,D\$	Nuts,in,May,!	Yes
	Nuts in May !	No

يمكن تنقيح البيانات التى يتم ادخالها كاستجابة لعبارة INPUT كما لو كانت مدخلات وقبل الضغط على مفتاح الادخال. وفيما يلى قائمة بالمفاتيح وخليط المفاتيح واجراءاتها.

المفتاح	الوظيفة
السهم الايمن أو Ctrl-\	ينقل نقطة البداية خانة واحدة إلى اليمين.
السهم الأيسر أو Ctrl-J	ينقل نقطة البداية خانة واحدة إلى اليسار.
سهم أيمن أو Ctrl-F	ينقل نقطة البداية كلمة واحدة لليمين.
سهم أيسر أو Ctrl-B	ينقل نقطة البداية كلمة واحدة لليسا.
Ctrl-K أو Home	ينقل نقطة البداية إلى بداية السطر.
Ctrl-N أو End	ينقل نقطة البداية نهاية السطر.
Ctrl-R أو Ins	يغير حالة الادخال من on إلى off والعكس. وعندما تكون حالة الادخال فى وضع on فتضاف الرموز التى يتم الضغط عليها إلى سطر المدخلات وعندما تكون فى وضع off فتتمحو الرموز التى يتم الضغط عليها على الحروف السابقة.
Ctrl-I أو Tab	يحرك نقطة البداية بخطوة جنولية إلى اليمين.
Del	لحذف الرمز الذى يقع تحت نقطة البداية.
Ctrl-H أو Backspace	لحذف رمز على يسار نقطة البداية ونقل بقية السطر خانة واحدة إلى اليسار فاذا كانت نقطة البداية فى بداية سطر المدخلات فتتحذف الرمز الموجود تحتها.

المفتاح	الوظيفة
Ctrl-E أو Ctrl-End	لحذف سطر من عند نقطة البداية وحتى نهايته.
Ctrl-U أو Esc	لحذف محتويات سطر بغض النظر عن وضع نقطة البداية.
Ctrl-M أو Enter	إنهاء المدخلات وتخزينها.
Ctrl- T	لتغيير وضع عرض الوظائف الموجود في قاعدة الشاشة من وضع on إلى وضع off والعكس.
Ctrl-C أو Ctrl-Break	لإنهاء المدخلات والخروج من البرنامج.

التطبيقات

عبارة INPUT هي طريقة سهلة ومتعددة الجوانب لقبول مدخلات من لوحة المفاتيح اثناء تنفيذ البرنامج. وبساطة تكوين العبارة مع قوته يسمح بالابتكارية في البرمجة. ويمكن أن تقبل المدخلات على هيئة عناصر سلاسل أو عناصر عددية أو منظومات أو سجلات، وفيما يلي قائمة أمثلة.

مثال ١

```
INPUT "Enter full name [First, Middle, and Last] ", First$, Middle$, Last$
```

يوجد أكثر من متغير واحد في قائمة متغيرات هذا المثال، ويجب أن تكون المدخلات عبارة عن ثلاث قيم مفصولة عن بعضها البعض بواسطة فواصل.

مثال ٢

```
INPUT "Enter current sales tax ", ST!
CONST MaxRow% = 20
DIM SampleArray(MaxRow%)
FOR Cnt = 1 TO MaxRow%
    INPUT "Next array value ", SampleArray[Cnt]
NEXT Cnt
```

يقبل هذا المثال مدخلات مباشرة في عناصر المنظومة SampleArray والمعرفة بأنها بها 20 عنصر.

```

TYPE Client
    Name AS STRING*40
    Company AS STRING*40
    Address AS STRING*80
END TYPE
DIM NextClient AS Client

INPUT "Client name and company ", NextClient.Name, NextClient.Company
    
```

يقبل هذا المثال مدخلات في نوع سجل يعرفه المستخدم، وتقبل المدخلات داخل كل عنصر بدلاً من قبولها داخل السجل ككل.

وتوضح الأمثلة التالية استخدام الفاصلة المنقوطة والفاصلة في عبارة INPUT واختلافات أخرى.

```

INPUT : "Type something ", X$
INPUT " Now type something else ! ", X$

INPUT : X$
INPUT " Continue ", X$

INPUT "What is your age" Age%
    
```

عملية تقليدية

استخدام عبارة INPUT موضح في البرنامج التالي، ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls F1=Help
Untitled

CONST MaxFile% = 10
TYPE FileRec
    FileName AS STRING * 15
    FileRef AS STRING * 40
END TYPE
DIM FileList(MaxFile%) AS FileRec

CLS
PRINT "Filename": TAB(40): "File Reference": PRINT STRINGS(60, "-")
FOR Cnt = 1 TO MaxFile%
    LOCATE Cnt + 2, 1: INPUT FileList(Cnt).FileName
    LOCATE Cnt + 2, 40: INPUT FileList(Cnt).FileRef
NEXT

Immediate

Run: [Untitled] Context: Program not running 0001:012
    
```

٢ - نفذ البرنامج ولاحظ استخدام عبارة INPUT وكيفية تحكمها في طريقة ادخال المدخلات في المنظومة من النوع الذى يعرفه المستفيد والمسماة FileRec. ادخل اسماء ملفات واوصافها كعينة واضغط على Ctrl-Break لإنهاء البرنامج.

Filename	File Reference
? Input.Bas	? Example INPUT statement
? Record.Inc	? Record defs. include file
? Movie.Bas	? Moving pictures program
? Sing.Bas	? Music program
? Jingle.Bas	? Music program
?	

٣ - اضغط على أى مفتاح للعودة إلى البرنامج. اضغط على Alt-F ثم اضغط على مفتاح الادخال واكتب N لإخلاء الشاشة.

٤ - انتقل إلى الدرس الرابع والسبعين للاستمرار في تسلسل التعلم.

الدرس الثالث والستون

عبارة INPUT#

الوصف

تؤدي عبارة INPUT# نفس الشيء مع البيانات التي يتم ادخالها من ملف والذي تؤدي عبارة INPUT مع البيانات التي يتم ادخالها من لوحة المفاتيح. تقرأ عبارة INPUT# بيانات من ملف تتابعي وتحدد البيانات للمتغيرات. وتكوينها هو كما يلي :

```
INPUT # filenum, variable list
```

جزء filenum هو رقم الملف المحدد بواسطة عبارة OPEN. وجزء variable list هو قائمة بأسماء المتغيرات التي تحدد البيانات المقروءة لها. يجب أن تتفق أنواع أسماء المتغيرات مع البيانات التي تقرأ فيها. ولا تطبع عبارة INPUT# علامة استفهام أثناء التنفيذ. وتكون عناصر البيانات المقروءة مرتبة في الملف بنفس الطريقة التي تدخل بها كاستجابة لعبارة INPUT#. ولمعرفة تفاصيل أكثر لكيفية ترتيب البيانات ارجع إلى الدرس الثاني والستون. عند قراءة قيم عديدة تهمل الفراغات الزائدة ورموز عودة العربة وتغذية السطر وتبدأ القيمة عندما يظهر رمز غير هذه الرموز وتنتهي بظهور فراغ أو فاصلة أو رمز عودة العربة أو رمز تغذية السطر. وعند قراءة قيم سلاسل تهمل الفراغات الزائدة ورموز عودة العربة وتغذية السطر. وعندما تظهر نهاية الملف أثناء قراءة قيمة سلسلة فيعتبر أن عنصر البيانات قد انتهى.

التطبيقات

عبارة INPUT# هي وسيلة مفيدة في قراءة بيانات من ملف بسرعة وتحديد المتغيرات في نفس الوقت. واستخدام عبارة INPUT# مع ملف بيانات مرتبة منتشر. وفيما يلي بعض الأمثلة.

```
OPEN "Score.Dat" FOR INPUT AS #3
INPUT #3 Test1, Test2, Test3

OPEN "Ledger" FOR RANDOM AS #2
FIELD #2 12 AS AcctNo, 25 AS AcctName$, 12 AS Amt
INPUT #2 AcctNo, AcctName$, Amt
```

عملية تقليدية

توضح هذه العملية عبارة INPUT#. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Call... FI-Help
<Untitled>
This program illustrates the use of the INPUT # statement. The program
opens the file created in Module 69, and lists it on the screen.

CLS
OPEN "Print.Fil" FOR INPUT AS #3
CLS
DO WHILE NOT EOF(3)
    INPUT #3, AS
    PRINT AS
LOOP
CLOSE #3

Immediate

Main: <Untitled> Context: <Untitled> 00005-012

```

٢ - نفذ البرنامج. لاحظ استخدام عبارة INPUT# في البرنامج. لاحظ بصفة خاصة عبارة IN-PUT# التي تقرأ السطر من الملف وحتى الفاصلة والتي تعتبر كنهاية للبيانات. يتسبب ذلك في أن تقوم عبارة INPUT# بقراءة البيانات بعد الفاصلة (في حقل amount) كسجل منفصل.

```

Earth Moving Equipment      12      $120
000
Farm Equipment              22      $85
000
Farm 22      $85
0
SOS

Press any key to continue

```

٣ - اضغط على أى مفتاح للعودة إلى البرنامج. اختر New من قائمة File واكتب N لإخلاء الشاشة.

٤ - انتقل إلى الدرس الثامن والستين للاستمرار فى تسلسل التعلم.

الدرس الرابع والستون

عبارة INPUT\$

الوصف

تقرأ عبارة INPUT\$ سلسلة من ملف، وتكوينها هو كما يلي :

```
INPUT$(NumOfChars, #Filename)
```

يحدد جزء NumOfChars عدد الرموز التي تقرأ من الملف، ويحدد جزء Filename الملف الذي يقرأ. ويجب أن يكون رقم الملف هو نفس الرقم المستخدم في عبارة OPEN لهذا الملف. وعندما يكون الملف المحدد مفتوحاً كملف اتصال عشوائي فيجب أن يكون NumOfChars أقل من طول السجل أو يساويه والقيمة التقليدية هي 128 رمزاً. وعندما يحدد أن الاتصال يتم بالملف كملف ثنائي فيجب أن يكون Num Of Chars أقل من أو يساوي 32,767.

إذا حذف جزء Filename فتستخدم وحدة المدخلات النمطية في إدخال البيانات ويكون أن تكون المدخلات قد أعيد توجيهها فتستخدم لوحة المفاتيح كوحدة مدخلات نمطية. أرجع إلى دليل DOS لمناقشة إعادة توجيه المدخلات والمخرجات. ولا تصدر عبارة INPUT\$ صدى للمدخلات على الشاشة.

التطبيقات

عبارة INPUT\$ مفيدة جداً في قراءة تسلسل من الرموز من ملف أو من وحدة، والمقدرة على قبول مدخلات من وحدة مدخلات نمطية هي ميزة إضافية. وفيما يلي بعض الأمثلة :

مثال ١

```
OPEN "Story.Txt" FOR INPUT AS #1
PRINT INPUT$(128, #1)
```

مثال ٢

```
Line$ = INPUT$(255)
```

يقبل المثال السابق مدخلات من لوحة المفاتيح في متغير السلسلة Line\$.

عملية تقليدية

هذه العملية توضح استخدام عبارة INPUT\$. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls
<Untitled>
' This program demonstrates the use of the INPUT$ statement.
' The program requests a filename and lists that file using the
' INPUT$ statement.
CLS
FILES "7777.BAS"
INPUT "Enter file to list: "; FileName$

IF FileName$ <> "" THEN
  OPEN FileName$ FOR INPUT AS #1
  InCh$ = INPUT$(1, #1)
  DO WHILE NOT EOF(1)
    IF (InCh$ <> CHR$(13)) THEN PRINT InCh$:
    InCh$ = INPUT$(1, #1)
  LOOP
  CLOSE #1
ELSE
  PRINT : PRINT "Thank you for participating ..."
END IF

```

----- Immediate -----

Run: <Untitled> Context: Program not running 00018:007

٢ - نفذ البرنامج. لاحظ استخدام عبارة INPUT\$ في البرنامج. لاحظ كذلك أن محتويات القرص لا تكون متطابقة مع شكل الشاشة التالي. اختر ملفاً نصياً لعرضه وادخل اسم الملف عند الملقن. اضغط على أى مفتاح للعودة إلى البرنامج.

```

C:\QB
BOX .BAS BOX2 .BAS CASE .BAS ASC .BAS
ABS .BAS PLAY .BAS OPEN .BAS
884736 Bytes free
Enter file to list: 7 asc.bas
Num$ = "1234": Num% = 0
FOR i = 1 TO LEN(Num$)
  Num% = Num% + ((ASC(MID$(Num$, i, 1)) - 48) * (10 ^ (LEN(Num$) - i)))
NEXT i
PRINT Num$, Num%

Press any key to continue

```

٣ - اختر New من قائمة File واكتب N لإخلاء الشاشة.

٤ - انتقل إلى الدرس السابع والسبعين للاستمرار في تسلسل التعلم.

الدرس الخامس والستون

دالة INSTR

الوصف

تعيد دالة INSTR موقع أول حدوث لسلسلة داخل سلسلة أخرى، وتكوينها هو كما يلي :

```
INSTR( start, string expression1, string expression2
```

الوصف	الجزء
كلمة من كلمات بيسك السريع المحجوزة.	INSTR
مؤشر اختياري يحدد موقع بداية البحث، فإذا لم يتحدد الموقع فيبدأ البحث عند الموقع 1.	start
السلسلة التي يجري فيها البحث، ويمكن أن تكون تعبير سلسلة أو متغير أو ثابت.	string expression 1
السلسلة التي يجري عنها البحث ويمكن أن تكون تعبير سلسلة أو متغير أو ثابت.	string expression 2

التطبيقات

يمكن استخدام عبارة INSTR لفحص داخل محتويات سلسلة بدون استخلاص سلسلة جزئية، والأمثلة التالية توضح بعض طرق استخدامها والنتائج.

مثال ١

```
FStr$ = "12345.0909"
PRINT INSTR(FStr$,".")
PRINT INSTR(6,FStr$,".")
```

مثال ٢

```
xstr$ = "Peace on earth and good will among men.": spos = 1
CLS
11:
spos = INSTR(spos + 1, xstr$, " ")
PRINT spos: " ":
IF spos > 0 THEN GOTO 11
```

عملية تقليدية

العملية التالية توضح استخدام دالة INSTR. ابدأ بتحميل ببسك السريع.

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls F1-Help
<Untitled>
' This program demonstrates the INSTR function
DIM A(26) AS INTEGER
Chcnt = 0: CLS
Istr$ = "CAESAR SIPPED HIS SHIFTER AND SEIZED HIS KNEES AND SNEEZED"
Istr2$ = Istr$
' Count the number of times each character occurs
' Use the ASCII values for the loop count
FOR c2 = 1 TO 26
  Chcnt = 0: SPos = 1 'set the starting position for INSTR

  FOR Cnt = 1 TO LEN(Istr$)

    IF SPos > 0 THEN
      SPos = INSTR(SPos, Istr$, CHR$(c2 + 64))
    END IF

    ' If character found then
    IF SPos > 0 THEN
      A(c2) = A(c2) + 1
      'replace the character already found with a non-relevant char.
      MID$(Istr$, SPos) = "x"
      'Display the progress with the string
      LOCATE 1, 1: PRINT Istr$
    END IF

  NEXT Cnt

NEXT c2

' Print the character count
c = 0
PRINT : PRINT Istr2$
PRINT : PRINT "The character count for this line is as follows: "
FOR Cnt = 1 TO 26
  ' If five counts have been printed, proceed to a new line
  IF c = 5 THEN PRINT : c = 0
  PRINT A(Cnt); "   "; CHR$(64 + Cnt); "'s ":
  c = c + 1
NEXT Cnt

```

Immediate

Main: <Untitled> Context: Program not running 00054:011

٢ - نفذ البرنامج ولاحظ استخدام دالة INSTR في البرنامج ولاحظ المخرجات.

[illegible]

CAESAR SIPPED HIS SNIFTER AND SEIZED HIS KNEES AND SNEEZED

The character count for this line is as follows:

4	A's	0	B's	1	C's	5	D's	10	E's
1	F's	0	G's	2	H's	5	I's	0	J's
1	K's	0	L's	0	M's	5	N's	0	O's
2	P's	0	Q's	2	R's	0	S's	1	T's
2	U's	0	V's	0	W's	0	X's	0	Y's
2	Z's								

Press any key to continue

- ٣ - اضغط على أى مفتاح للعودة إلى البرنامج. من قائمة File اختر Save واكتب INSERT, BAS كاسم للملف وحدد أن شكل الملف نصي واحفظ هذا البرنامج.
- ٤ - من قائمة File اختر New.
- ٥ - انتقل إلى الدرس المائة والرابع عشر للاستمرار فى تسلسل التعلم.

الدرس السادس والستون

دالة INT

الوصف

تعيد دالة INT أقصى قيمة عددية صحيحة مساوية لمؤشر التعبير العددي أو أقل منه، وتكوينها هو كما يلي :

`INT(numeric expression)`

يعود التعبير العددي numeric expression بعد حذف الكسر العشري منه ومع وضع الإشارة.

التطبيقات

دالة INT هي إحدى نوال التقريب المتاحة في البيسك السريع وتستخدم في الحصول على أقرب قيمة عددية صحيحة محددة اشارتها لا تكون أكبر من التعبير العددي، وفيما يلي بعض الأمثلة :

`PRINT INT(12.5)`

المخرجات : 12

`PRINT INT(-7.33)`

المخرجات : -8

`PRINT INT(99.31)`

المخرجات : 99

`PRINT INT(-823.001)`

المخرجات : -824

عملية تقليدية

العملية التالية توضح استخدام دالة INT، ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :

```
File Edit View Search Run Debug Calls Window Help
<Untitled>
This program demonstrates the INT function
a = 2.8: b = a + .5: c = -a
d = c + .5
PRINT a, b, c, d
PRINT INT(a), INT(b), INT(c), INT(d)

Immediate

Main: <Untitled> Context: Program not running 80005-837
```

٢ - نفذ البرنامج ولاحظ استخدام دالة INT في البرنامج.

2.8	3.3	-2.8	-2.3
2	3	-3	-3

Press any key to continue

٣ - اضغط على أى مفتاح للعودة إلى البرنامج. من قائمة File اختر New مع عدم حفظ هذا البرنامج.

٤ - انتقل إلى الدرس الخامس للاستمرار فى تسلسل التعلم.

الدرس السابع والستون

عبارة IOCTL ودالة IOCTL\$

الوصف

تستخدم عبارة IOCTL ودالة IOCTL\$ فى التداخل مع مشغلات الوحدات وتكوينها هو كما يلى:

```
IOCTL #file num. string
IOCTL$ (#file num)
```

ترسل عبارة IOCTL سلسلة بيانات تحكم إلى مشغل الوحدة المحددة. جزء #file num هو رقم الملف المستخدم فى عبارة OPEN لفتح الوحدة. وجزء string هو بيانات التحكم التى ترسل إلى مشغل الوحدة كإمر. ويمكن أن يصل طول السلسلة حتى 32,767 رمزاً كحد أقصى.

وتستقبل دالة IOCTL\$ سلسلة بيانات تحكم من مشغل وحدة معين. وجزء #file num هو نفسه مثل ما هو موجود فى عبارة IOCTL.

وفيما يلى المتطلبات اللازمة لعمل IOCTL و IOCTL\$.

- يجب أن يكون مشغل الوحدة معداً.
 - يجب أن يكون مشغل الوحدة قادراً على تشغيل سلاسل IOCTL، ويمكن الحصول على هذه المعلومات من الوثائق الخاصة بمشغل الوحدة. كما يمكن الحصول كذلك على معلومات عن دعم IOCTL من خلال إزعاجات DOS (DOS interrupts). وللمزيد من المعلومات عن استدعاءات دوال DOS وإزعاجاته ارجع إلى الدليل التقنى لنظام DOS.
 - يجب أن يؤدى بيسك السريع عبارة OPEN ويجب أن يظل الملف مفتوحاً.
- وحدات بيسك السريع : LPT1 و COM1 و COM2 و SCRIN: و CONS: ووحدات مجموعة DOS من : A حتى Z: لا تدعم IOCTL.

التطبيقات

تستخدم عبارة `IOCTL` ودالة `IOCTL$` في تطبيقات برمجة مشغلات الوحدات ومثل هذه البرمجة تكون محددة بوحدة معينة. ومناقشة مشغلات الوحدات تقع خارج مدى هذا الكتاب. انتقل إلى الدرس الأربعين للاستمرار في تسلسل التعلم.

الدرس الثامن والستون

عبارات KEY

الوصف

يتعامل هذا الجزء مع عبارات المفاتيح KEY المستخدمة في تحديد قيم لمفاتيح الوظائف وإنتاج تسلسلات مفاتيح يعرفها المستفيد وسرد تحديدات المفاتيح وتمكين وإلغاء إمكانية وإيقاف تصيد أحداث KEY. والعبارات لها التكوين التالي :

```
KEY n, string
KEY LIST
KEY ON
KEY OFF
KEY (n) ON
KEY (n) OFF
KEY (n) STOP
```

عبارة KEY n, string : تحدد هذه العبارة تعبير سلسلة لمفتاح وظيفة معين، وجزء n هو عدد يقع بين 1 و 10 أو 30 أو 31، تعني الأعداد من 1 إلى 10 مفاتيح الوظائف من F1 حتى F10، كما أن 30 و 31 يحددان مفتاحي الوظائف F11 و F12 في لوحة مفاتيح AT الموسعة. جزء string هو متغير سلسلة أو ثابت يحدد إلى مفتاح معين، يتسبب ذلك في إنتاج السلسلة عندما يكون المفتاح مضغوطاً عليه، ويمكن تحديد قيمة السلسلة بحد أقصى 15 رمزاً فإذا كان طول السلسلة أكبر من ذلك فتعمل الرموز الزائدة عن 15 رمزاً، وتحديد سلسلة فارغة لمفتاح معين يجعل المفتاح غير قادر على أداء التحديد السابق له، وتسمى هذه التحديدات بالمفاتيح الناعمة soft keys.

عبارة KEY LIST : تتسبب هذه العبارة في طباعة تحديدات السلاسل لمفاتيح الوظائف على الشاشة. وتعرض كل الرموز (15 رمزاً) الموجودة في السلسلة والمحددة لكل مفتاح على الشاشة.

عبارة KEY ON : تتسبب هذه العبارة في عرض تحديدات المفاتيح الناعمة في قاعدة الشاشة، ويعرض أول ستة رموز من السلسلة فقط.

عبارة KEY OFF : تتسبب هذه العبارة في حذف عرض KEY ON من على الشاشة لجعل هذا السطر من الشاشة متاحاً للبرنامج.

عبارات اصطلياد نشاط المفتاح EVENT TRAPPING KEY STATEMENTS
اصطلياد نشاط المفتاح متاح لمفاتيح الوظائف من 1 إلى 12 ولمفاتيح التحكم فى نقطة البداية
والمفاتيح التى يعرفها المستفيد وهى 10. قيمة الجزء n هى كما يلى :

المفاتيح	n
مفاتيح الوظائف من 1 إلى 10.	1-10
حركة نقطة البداية لأعلى.	11
حركة نقطة البداية لليساار.	12
حركة نقطة البداية لليمين.	13
حركة نقطة البداية لأسفل.	14
المفاتيح التى يعرفها المستفيد.	15-25
مفتاحى F11 و F12.	30,31

وفىما يلى التكوين المستخدم لتعريف المفاتيح التى يحددها المستفيد :

KEY n, CHR\$(kbd flag) + CHR\$(scan code)

حيث n هو رقم يتراوح من 15 إلى 25 وجزء kbd flag هو رمز ينتج عندما يتم الضغط على
مفتاح متسع وجزء scan code هو رمز فحص فى لوحة المفاتيح لمفتاح محدد. ويسمح هذا
بتعريف اكتشاف واصطلياد الخليط من مشاوير المفاتيح. وفىما يلى قائمة بقيم إشارة kbd التى
يمكن استخدامها فى الخليط.

المفتاح	kbd flag
لاتوجد إشارة لوحة مفاتيح	&h00
مفاتيح ترحيل يسار أو يمين	&h01-&h03
مفتاح تحكم	&h04
مفتاح تبديل	&h08
مفتاح اغلاق الأعداد	&h20
مفتاح اغلاق الحروف الصغيرة	&h40
مفتاح لوحة مفاتيح AT المتسعة	&h80

وفيما يلي قائمة برموز فحص لوحة المفاتيح، لاحظ أن الرموز موجودة بالنظام السادس عشر.

Key	Code	Key	Code	Key	Code
Esc	01	Ctrl	1D	Spacebar	39
!,1	02	A	1E	Capslock	3A
@,2	03	S	1F	F1	3B
#,3	04	D	20	F2	3C
\$,4	05	F	21	F3	3D
%,5	06	G	22	F4	3E
^,6	07	H	23	F5	3F
&,7	08	J	24	F6	40
*,8	09	K	25	F7	41
(,9	0A	L	26	F8	42
),0	0B	;;,	27	F9	43
_,-	0C	~,'	28	F10	44
+ , =	0D	~,'	29	Numlock	45
Left	0E	Left Shft	2A	ScrLck	46
Tab	0F	!,\	2B	Home,7	47
Q	10	Z	2C	Up,8	48
W	11	X	2D	PgUp,9	49
E	12	C	2E	-	4A
R	13	V	2F	Left,4	4B
T	14	B	30	5	4C
Y	15	N	31	Right,6	4D
U	16	M	32	+	4E
I	17	< ,	33	End,1	4F
O	18	> ,	34	Down,2	50
P	19	? , /	35	PgDn,3	51
{ , [1A	Right Shf	36	Ins,0	52
} ,]	1B	PrtScr,*	37	Del,.	53
Return	1C	Alt	38		

عبارة KEY (n) ON: تمكن هذه العبارة اصطلياد النشاط للمفتاح n. وعندما يكون هذا المفتاح مضغوطاً عليه فيميزه البرنامج ويمكن أن يتأثر به.

عبارة KEY (n) OFF: تلغى هذه العبارة من مقدرة اصطلياد النشاط للمفتاح n. وتسبب في ألا يميز البرنامج متى يكون هذا المفتاح مضغوطاً أثناء تنفيذ البرنامج. ويمكن على أية حال اكتشاف المفتاح باستخدام طرق أخرى.

عبارة KEY (n) STOP: توقف هذه العبارة اصطلياد نشاط المفتاح n. والأنشطة التي تنفذ بعد هذه العبارة لا يحدث لها اصطلياد لكن يمكن تذكرها. بعد تنفيذ عبارات KEY (n) ON متتالية يتم تشغيل النشاط.

التطبيقات

تقدم عبارات KEY طريقة قوية وشاملة للتعامل مع مدخلات لوحة المفاتيح. وفي أى تطبيق متداخل مطور يوصى بشدة باستخدام هذه العبارات. وفيما يلي أمثلة لعبارات KEY :

```
KEY ON
KEY 1, "Help"
KEY 2, "Main Menu"
KEY 3, "Create"
KEY 4, "Edit"
KEY 5, "Browse"
CLS
KEY LIST
INPUT t$
```

وعندما تنفذ هذه العبارات ينتج عن تنفيذها شاشة تشبه ما يلي :

```
F1 Help
F2 Main Menu
F3 Create
F4 Edit
F5 Browse
F6
F7
F8
F9
F10
?
```

1 Help 2 Main Menu 3 Create 4 Edit 5 Browse 6 7 8 9 10

وقد كانت هناك عبارة INPUT بحيث يمكن ملاحظة الشاشة الناتجة. وفيما يلي أمثلة لعبارات KEY.

```
KEY 20, Chr$(&h0) + Chr$(&h01) 'define the Esc key
KEY 15, Chr$(&h05) + Chr$(&h01) 'define Ctrl-Shift-Esc
KEY (20) ON: KEY (15) ON 'enable event trapping for both
ON KEY (20) GOSUB Abort
ON KEY (15) GOSUB ChangeModes:
..
KEY (20) OFF 'disable the Esc key
RETURN
```

عملية تقليدية

هذه العملية عبارة عن توضيح موجز لعبارات KEY (n). يعرف البرنامج مشوارى مفاتيح Ctrl-A و Alt-A ويمكن من اصطيد النشاط لكل من المفاتيح ويوضح أنهما يعملان.

ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :

```

FILE Edit View Search Run Debug Calls F1=Help
Untitled
'Define the key Ctrl-A
KEY 15, CHR$(&H4) + CHR$(&H1E)
'Define the key Alt-A
KEY 16, CHR$(&HB) + CHR$(&H1E)
KEY(15) ON: KEY(16) ON
ON KEY(15) GOSUB CtrlA
ON KEY(16) GOSUB Alta

CLS
PRINT "This program illustrates the use of KEY(n) statements."
PRINT "Two keys are defined (Ctrl-A, Alt-A) and event trapping "
PRINT "is enabled. Test it."

DO
  Ch$ = INKEY$
  LOOP UNTIL (Ch$ <> "")

CtrlA:
  PRINT "Ctrl-A detected"
  RETURN

Alta:
  PRINT "Alt-A detected"
  END

```

Immediate

Menu: vUntitled * Load... Printout Quit

٢ - نفذ البرنامج. اضغط على Ctrl-A و Alt-A لفصل البرنامج. لاحظ استخدام عبارات KEY (n) في البرنامج. وفيما يلي عينة للتنفيذ :

```
This program illustrates the use of KEY(n) statements.  
Two keys are defined (Ctrl-A, Alt-A) and event trapping  
is enabled. Test it.  
Ctrl-A detected  
Alt-A detected
```

Press any key to continue

٣ - ارجع إلى البرنامج داخل الشاشة دون أن تحفظ هذا البرنامج.

٤ - انتقل إلى الدرس العاشر للاستمرار في تسلسل التعلم.

الدرس التاسع والستون

عبارة KILL

الوصف

تحذف عبارة KILL ملفاً محدداً من القرص. وتكوينها هو كما يلي :

KILL file specification

يقدم جزء file specification معلومات عن موقع الملف المطلوب ويشمل بصورة اختيارية المسار. ويجب أن تكون مواصفات الملف موضوعة بين علامتى تنصيص. ويمكن اتباع تكوين مسار DOS. ويمكن استخدام اسم الملف الموجود فى مواصفات الملف مع الرموز wild-card من DOS فى حذف أكثر من ملف واحد فى نفس الوقت. فإذا ما حاولت أن تحذف "KILL" ملفاً مفتوحاً كمدخلات أو كمخرجات فينتج بيسك السريع رسالة خطأ تفيد بأن الملف مفتوح بالفعل.

التطبيقات

تستخدم عبارة KILL فى حذف ملفات أثناء تنفيذ البرنامج. وهى مفيدة بصفة خاصة عندما ينتج البرنامج ملفات مؤقتة أثناء التنفيذ. وفيما يلى بعض الأمثلة :

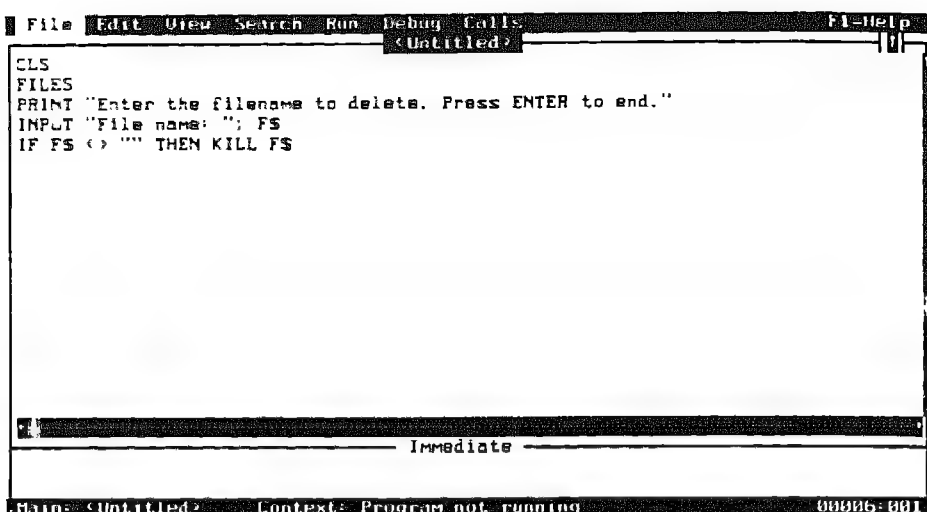
```
KILL "SCORE.DAT"
KILL "C:\MASM\CLRSCR.BAK"
KILL "C:\C\*.BAK"
KILL "SCORE?.*"
KILL "SCORE*.*"
```

لاحظ الفرق فى الرموز الخاصة بين آخر مثال والمثال السابق له. مواصفة الملف "SCORE?.*" تتسبب فى حذف كل الملفات التى تبدأ بكلمة SCORE يليها أى رمز آخر أو أى عدة رموز أخرى ولها أى اتساع.

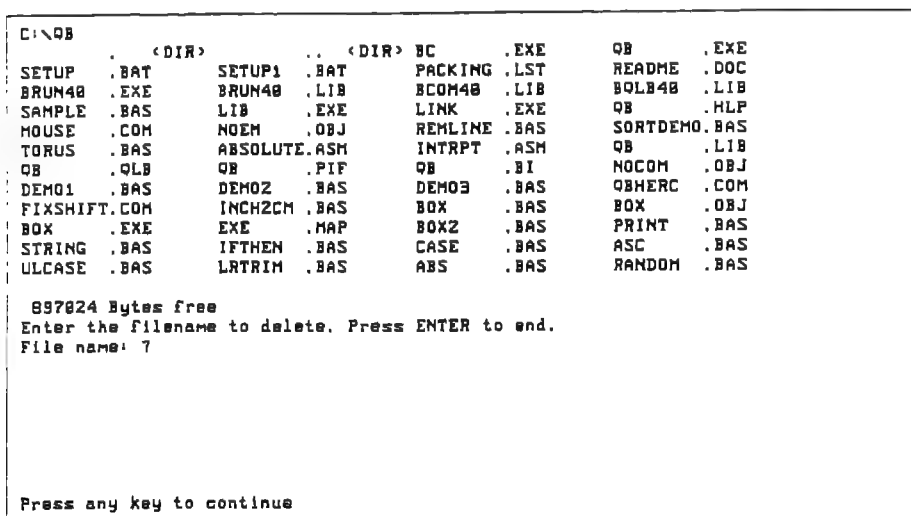
عملية تقليدية

توضح العملية التالية استخدام عبارة KILL. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :



٢ - نفذ البرنامج. لاحظ استخدام عبارة KILL في البرنامج. اضغط على مفتاح الإدخال لإنهاء البرنامج بون أن تحذف أى ملف.



٣ - اضغط على أى مفتاح للعودة إلى البرنامج. اختر New من قائمة File واكتب N لإخلاء الشاشة.

٤ - انتقل إلى الدرس المائة والستين للاستمرار فى تسلسل التعلم.

الدرس السبعون

الاسماء LABELS

الوصف

الاسماء Labels هي أسماء تستخدم كمحددات لمواقع. وتستخدم في تعريف موقع أو سطر في البرنامج للاتصال به فيما بعد. وتستخدم أساساً بفرض عمل تفرعات. وأما أن تكون الاسماء في بيسك السريع عديدة أو حرفية عديدة. ولا يمكن للاسماء أن تحتوى على أى رموز غير الأعداد والحروف الهجائية أى إنها لا يمكن أن تحتوى على أى رموز خاصة.

يمكن أن يبدأ الاسم برقم أو بحرف ولا يمكن أن يزيد طوله عن 40 خانة وينتهى دائماً بنقطتين رأسيين. ويمكن ألا تتبع النقطتان الرأسيتان الاسم مباشرة. ويسمح بوجود اسم label واحد فقط على سطر واقعى واحد. وفيما يلي بعض الأمثلة :

```
ShowPrompt:
Menu01:
099:
FileKill:
```

التطبيقات

تستخدم الاسماء labels كأدلة لعبارات GOTO و GOSUB. ولا يمكن استخدامها مع عبارات IF..THEN. دليل تعريف السطر في عبارة IF..THEN يجب أن يكون رقم سطر. استخدام اسماء labels ذات معنى في برنامج البيسك يجعل الشفرة أسهل في قراءتها عن استخدام أرقام الأسطر.

عملية تقليدية

توضح هذه العملية أحد الاستخدامات الممكنة للاسماء labels. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```

File Edit View Search Run Debug Calls File Help
Untitled
CLS
Again:
INPUT "Enter selection (1..3)"; S
IF S = 1 THEN GOTO Choice1
IF S = 2 THEN GOTO Choice2
IF S = 3 THEN GOTO Choice3
GOTO Again

Choice1:
PRINT "Hello from label Choice1": END

Choice2:
PRINT "Hello from label Choice2": END

Choice3:
PRINT "Hello from label Choice3": END

```

Immediate

File Edit View Search Run Debug Calls File Help

٢ - اضغط على Shift-F5 لتنفيذ البرنامج. اكتب 2 واضغط على مفتاح الإدخال للحصول على المخرجات التالية :

```

Enter selection (1..3) 2
Hello from label Choice2

```

Press any key to continue

٣ - اضغط على أى مفتاح للعودة إلى قائمة البرنامج.

٤ - احفظ البرنامج فى شكل نصى تحت اسم LABEL.BAS. اضغط على Alt-F واكتب S لحفظ صندوق الحوار. اكتب LABEL.BAS كاسم للملف واضغط على Tab واستخدم مفاتيح الأسهم لتحديد أن الشكل نصى. اضغط على Tab مرة أخرى ثم اضغط على قسيب المسافات.

٥ - انتقل إلى الدرس التاسع والسبعين للاستمرار فى تسلسل التعلم.

الدرس الحادى والسبعون

دالة LBOUND و UBOUND

الوصف

تعيد دالة LBOUND أقل دليل لبعاد معين لمنظومة. وتعيد دالة UBOUND أكبر دليل لبعاد معين لمنظومة. والتكوين هو كما يلى :

```
LBOUND(array,dimension)
UBOUND(array,dimension)
```

جزء array هو متغير منظومة وجزء dimension هو البعد المراد تحديد أكبر أو أصغر دليل له. ويترك جزء dimension عندما يكون للمنظومة بعد واحد فقط. وتعيد دالة LBOUND القيمة 1 أو القيمة 0 بصورة تقليدية طبقاً لأعداد أساس الخيار OPTION BASE.

التطبيقات

تستخدم دوال LBOUND و UBOUND فى ايجاد الحدود الدنيا والعليا لبعاد المنظومة. وهذا مفيد فى تقويم حجم المنظومة وعندما لا تكون المنظومة موضحة محلياً وعندما لا يكون للبرنامج الفرعى اتصال بالتوضيحات. وفيما يلى بعض الأمثلة :

مثال ١

```
DIM Rt(100,2)
ArrLow = LBOUND(Rt,1)
ArrHigh = UBOUND(Rt,1)
```

مثال ٢

```
DIM Qt(12,10,99)
AL = LBOUND(Qt,2)
AL3 = LBOUND(Qt,3)
AH1 = UBOUND(Qt,1)
```

مثال ٣

```
OPTION BASE 1
DIM Sngl(26)
SnglLow = LBOUND(Sngl)
SnglHigh = UBOUND(Sngl)
```


عملية تقليدية

هذه العملية توضح استخدام دالتى LBOUND و UBOUND. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```
File Edit View Search Run Debug Calls F1=Help
<Untitled>
The following program demonstrates the use of the UBOUND and LBOUND
statements. The program loads two sets of array values and
finds the minimum and maximum values in those arrays.

DECLARE SUB FindMinMax (A%(), MinVal, MaxVal)

Max = 15
DIM A%(Max)
GOTO Start

LoadArray:
FOR Cnt = 1 TO Max
    READ A%(Cnt)
NEXT
RETURN

Start:
CLS
GOSUB LoadArray
PRINT "First pass"
CALL FindMinMax(A%(), MinVal, MaxVal)
READ Max
REDIM A%(Max)
GOSUB LoadArray
PRINT "Second pass"
CALL FindMinMax(A%(), MinVal, MaxVal)

DATA 12,23,33,43,1,56,98,656,323,44,5,88,67,54,18
DATA 18
DATA 8,89,76,54,23,32,12,4,33,54
```

```
File Edit View Search Run Debug Calls F1=Help
<Untitled>:FindMinMax
SUB FindMinMax (A%(), MinVal, MaxVal)
    MinVal = A%(1): MaxVal = A%(1)
    FOR Cnt = LBOUND(A%) + 1 TO UBOUND(A%)
        IF MinVal > A%(Cnt) THEN
            MinVal = A%(Cnt)
        END IF
        IF MaxVal < A%(Cnt) THEN
            MaxVal = A%(Cnt)
        END IF
    NEXT
    PRINT "Minimum value in array: "; MinVal, "Maximum value in array: "; MaxVal
END SUB

Immediate
Main: <Untitled> Context: Program not running 00011-070
```

٢ - نفذ البرنامج. لاحظ استخدام دالتى LBOUND و UBOUND فى البرنامج.

```
First pass
Minimum value in array: 1 Maximum value in array: 656
Second pass
Minimum value in array: 4 Maximum value in array: 89
```

Press any key to continue

٣ - ارجع إلى البرنامج، احفظ البرنامج على أنه برنامج نصي وله الاسم LBOUND.BAS مع اخلاء الشاشة.

٤ - انتقل إلى الدرس المائة والسابع والثلاثين للاستمرار في تسلسل التعلم.

الدرس الثاني والسبعون

دالتا \$LCASE و \$UCASE

الوصف

تحول دالتا \$LCASE و \$UCASE قيم سلاسل إلى الحالة السفلية (حروف صغيرة) والحالة العلوية (حروف كبيرة) على التوالي، وتكوين الدالتين هو كما يلي :

```
$LCASE$ String exp ;
$UCASE$ String exp ;
```

أجزاء string exp في كل من التكوينين هي تعبيرات سلاسل. تقبل الدالتان كل من السلاسل ثابتة الطول ومتغيرة الطول كمؤشرات لها. ويمكن أن يكون تعبير السلسلة ثابت سلسلة أو متغير سلسلة أو أي تعبير ينتج عنه سلسلة.

التطبيقات

نوال \$LCASE و \$UCASE تكون أكثر نفعاً في مقارنات سلاسل من المهام فيها تمييز حالة الحروف (سفلية أو علوية)، وفيما يلي بعض الأمثلة لاستخداماتهما :

مثال ١

```
DO WHILE $UCASE$(Choice$) <> "Q"
LOOP
```

مثال ٢

```
IF $UCASE$(Response$) = $UCASE$(Option$) THEN
END IF
```

مثال ٣

```
SELECT CASE $LCASE$(Entry$)
CASE "1"
END SELECT
```

عملية تقليدية

تعطى هذه العملية برنامجاً لتوضيح دالتى LCASE\$ و UCASE\$ لمقارنات سلاسل تهتم بالحالة. وتقبل الدالة المعرفة فى هذا البرنامج مدخلات وتقارنها مع الاختيارات المسموح بها وتعيد القيمة التى أدخلت إلى العبارة المنادية. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```

File Edit View Search Run Debug Calls F1=Help
ULCASE.BAS
DECLARE FUNCTION GetChar$ (ValidCh AS STRING)
'This program demonstrates the UCASE$ function

CLS
LOCATE 23, 1
PRINT "E)dit / C)reate / D)elate / Q)uit ?"
Selection$ = GetChar$("EDCQ")

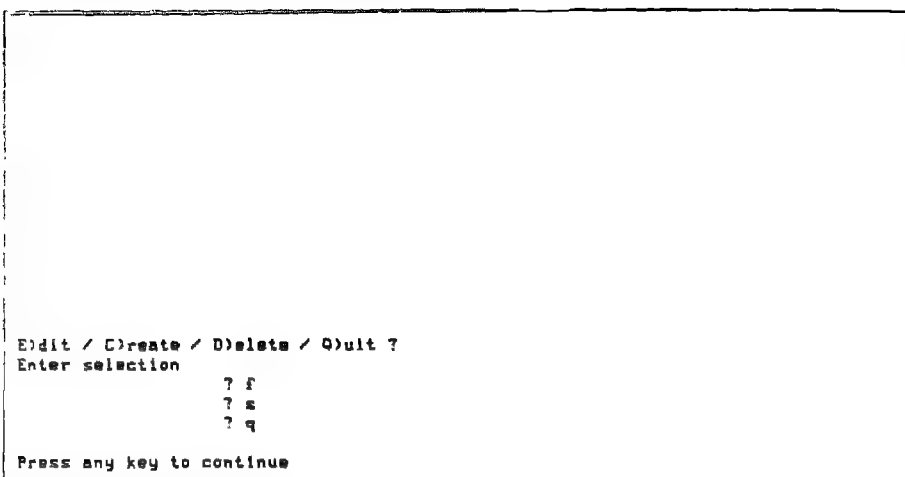
File Edit View Search Run Debug Calls F1=Help
ULCASE.BAS: GetChar
FUNCTION GetChar$ (ValidCh AS STRING)
DIM Choice AS STRING
LOCATE 24, 1: PRINT "Enter selection "
Choice = ""
DO WHILE INSTR(ValidCh, UCASE$(Choice)) = 0
LOCATE 24, 20: INPUT Choice
LOOP
GetChar$ = Choice
END FUNCTION

Immediate

Main: ULCASE.BAS Context: Program not running 00001:001

```

٢ - نفذ البرنامج ولاحظ سلوك البرنامج مع قبوله للمدخلات. ادخل حرفاً واحداً من الاختيار لإيقاف البرنامج.



٣ - اضغط على أى مفتاح للعودة إلى البرنامج.

٤ - من قائمة File اختر Save واكتب UCASE.BAS كاسم للملف. حدد أن شكل الملف هو نصي واحفظ الملف.

٥ - انتقل إلى الدرس السادس والثمانين للاستمرار فى تسلسل التعلم.

الدرس الثالث والسبعون

دالة LEFT\$

الوصف

تعيد دالة LEFT\$ عدد الرموز المحدد في أقصى اليسار من مؤشر تعبير السلسلة، وتكوينها هو كما يلي :

LEFT\$(String expression, num)

الوصف	الجزء
كلمة من كلمات بيسك السريع المحجوزة. السلسلة التي يعود منها العدد المحدد للرمز الموجود في أقصى اليسار، ويمكن أن يكون تعبير سلسلة أو متغير أو ثابت. عدد الرموز المطلوب اعادتها، فإذا كان num أكبر من طول السلسلة فتعود كل السلسلة.	LEFT\$ string expression num

التطبيقات

دالة LEFT\$ هي وسيلة أخرى لتشغيل سلاسل في بيسك السريع، وتستخدم في الحصول على سلسلة جزئية من تعبير سلسلة طبقاً لاتجاه محدد، ودائماً ما تكون السلسلة الجزئية التي تعود هي عدد الرموز المحدد في أقصى اليسار، وفيما يلي بعض الأمثلة :

مثال ١

```
Name$ = "Marmaduke Blenkinsop"
PRINT LEFT$(Name$.INSTR(Name$," "))
```

يبحث هذا المثال عن أول حدوث لفراغ ويطبع كل الرموز التي تسبق هذا الفراغ الأول.

مثال ٢

```
Phrase$ = "Eighty in time"
PRINT LEFT$ Phrase$,40
```

يطبع هذا المثال محتويات السلسلة Phrase\$ نظراً لأن عدد الرموز المحدد في قائمة
LEFT\$ أكثر من LEN (Phrase\$).

مثال ٣

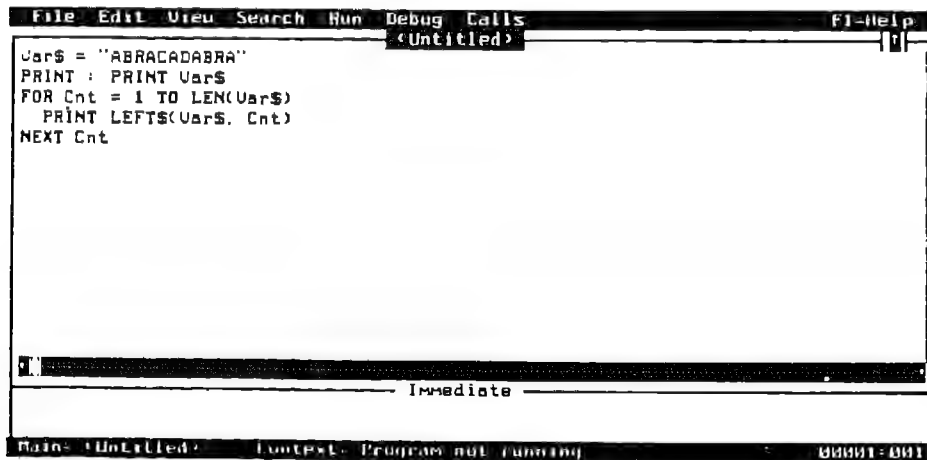
```
Phrase$ = "Saves nine"
PRINT LEFT$ Phrase$,0
```

تكون مخرجات هذا المثال عبارة عن سلسلة فارغة لأن عدد الرموز المحددة هو صفر.

عملية تقليدية

لتجربة برنامج بسيط يوضح دالة LEFT\$ استمر على النحو التالي. ابدأ بتحميل بيسك
السريع.

١ - اكتب البرنامج التالي :



٢ - نفذ البرنامج ولاحظ المخرجات واستخدام LEFT\$ في البرنامج.

```
ABRACADABRA
A
AB
ABR
ABRA
ABRAC
ABRACA
ABRACAD
ABRACADA
ABRACADAB
ABRACADABR
ABRACADABRA
```

Press any key to continue

٣ - اضغط على أى مفتاح للعودة إلى البرنامج. اضغط على Alt-F ثم اضغط على مفتاح الانخال واكتب N لإخلاء الشاشة.

٤ - انتقل إلى الدرس الثانى والسبعين للاستمرار فى تسلسل التعلم.

الدرس الرابع والسبعون

دالة LEN

الوصف

تعطى دالة LEN طول السلسلة المحددة أو عدد البايت اللازم للمتغير المحدد. وتكوينها هو كمايلي:

LEN 'String expression' OR LEN 'Variable'.

الوصف	الجزء
كلمة من كلمات بيسك المحجوزة. تعبير سلسلة أو ثابت حرفي. اسم متغير بيسك صحيح ويمكن أن يكون من أى نوع	LEN String expression variable

التطبيقات

يختلف استخدام دالة LEN. ففي المثال المقدم فى الدرس الثالث تستخدم LEN فى تضبيط النص فى منتصف الصندوق. وتشمل الاستخدامات الأخرى حدود عدادات الدورات وعمل تشكيلات المخرجات وعمل التشكيلات النصية. ويمكن لدالة LEN، عند استخدامها فى الحصول على بايت الذاكرة اللازم لأحد المتغيرات، ان تستخدم فى تقويم متطلبات تقويم ذاكرة البرنامج وحجم السجل للاتصال بملف عشوائى. وفيما يلى بعض الأمثلة :

مثال ١

```
PRINT "Long word is " LEN("Supercalifragilistic");  
      "chars. long"
```

مثال ٢

```
LStr$ = 'New kid on the block. Eh?'  
FOR Cnt = 1 TO LEN LStr$  
NEXT Cnt
```

يستخدم هذا المثال القيمة التي تعود من LEN كقيمة تحكم في دورة FOR.

مثال ٢

```
LStr$ = "Move it to the right!"
PRINT TAB(79-LEN(LStr$)) LStr$
```

مثال ٤

يعكس هذا المثال سلسلة مدخلات ويوضح أحد استخدامات دالة LEN.

```
LStr$ = "ABLE WAS I ERE I SAW ELBA"
PRINT LStr$
FOR SPos = LEN(LStr$) TO 1 STEP -1
  PRINT MID$(LStr$,SPos,1);
NEXT SPos
```

عملية تقليدية

يوضح البرنامج التالي كيفية ايجاد البايت اللازمة بواسطة متغير. ابدأ بتحميل بيسك

السريع.

١ - اكتب البرنامج التالي :

```
File Edit View Search Run Debug Call... File Help
<Untitled>
CLS
TYPE Customer
  LName AS STRING * 40
  Address AS STRING * 40
  LastInvoice AS STRING * 40
  Outstanding AS DOUBLE
END TYPE

DIM CreditPeriod AS INTEGER, Interest AS SINGLE
DIM NextCust AS Customer

PRINT "Credit Period needs "; LEN(CreditPeriod); " bytes."
PRINT "Interest needs "; LEN(Interest); " bytes."
PRINT "Customer record needs "; LEN(NextCust); " bytes."

Immediate

Main: <Untitled> Code: C: Program Files\VB\VB60\VB60.exe
```

٢ - اضغط على Shift-F5 لتنفيذ البرنامج. لاحظ استخدام دالة LEN في الحصول على حجم البايث لمتغيرات يعرفها المستخدم.

```
Credit Period needs 2 bytes.  
Interest needs 4 bytes.  
Customer record needs 128 bytes.
```

Press any key to continue

٣ - اضغط على أى مفتاح للعودة إلى البرنامج. اضغط على Alt-F ثم اضغط على مفتاح الإدخال واكتب N لإخلاء الشاشة.

٤ - انتقل إلى الدرس السابع والثمانين للاستمرار في تسلسل التعلم.

الدرس الخامس والسبعون

عبارة LET

الوصف

تستخدم عبارة LET فى تحديد قيمة لمتغير، ويمكن أن تكون القيمة تعبيراً أو ثابتاً ويجب أن تكون من نفس نوع المتغير. وتكوينها هو كما يلى :

LET variable-expression

كلمة بيسك السريع LET المحجوزة تكون اختيارية فى التكوين وتكفى علامة التساوى. تحدد قيمة التعبير الموجود فى الطرف الأيمن لاسم المتغير الموجود فى الطرف الأيسر. والمتغير هو أى متغير بيسك سريع صحيح ويمكن للتعبير أن يكون عددياً أو سلسلة أو حرفياً. فى حالة متغيرات من النوع الأساسى أو الأولى فإذا كان المتغير يشار إليه لأول مرة فى البرنامج فيلزم للإشارة أن توضح المتغير. ويصبح المتغير نشطاً فى أول مرة تستخدمه. ويعمل التحديد بطريقة طيبة حتى إذا لم يستخدم الفعل LET. والتحديد A=10 صحيح من ناحية التكوين ولاينتج البرنامج خطأ أثناء ترجمته وينفذ دون أن يتأثر بعدم وجود الفعل LET.

التطبيقات

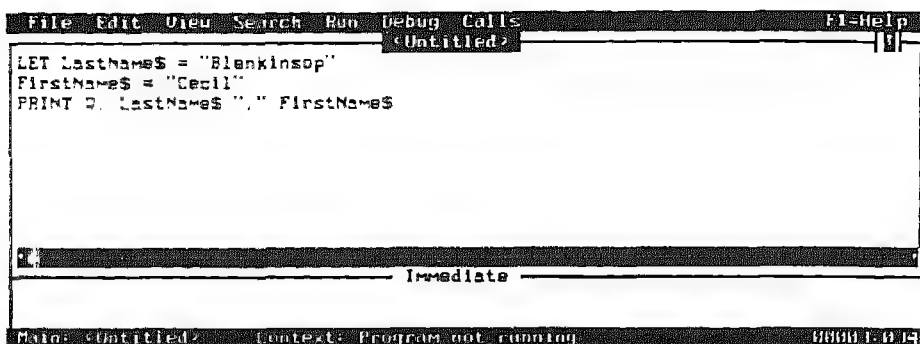
استخدم LET فى تعديل قيمة المتغير الحالية أو فى وضع قيمة ابتدائية معينة للمتغير. وفيما يلى بعض الأمثلة :

```
LET A = 23: LET B = 15
LET Question$ = 'What's up, Doc?'
NewQuestion$ = "What's up..DUCK?"
```

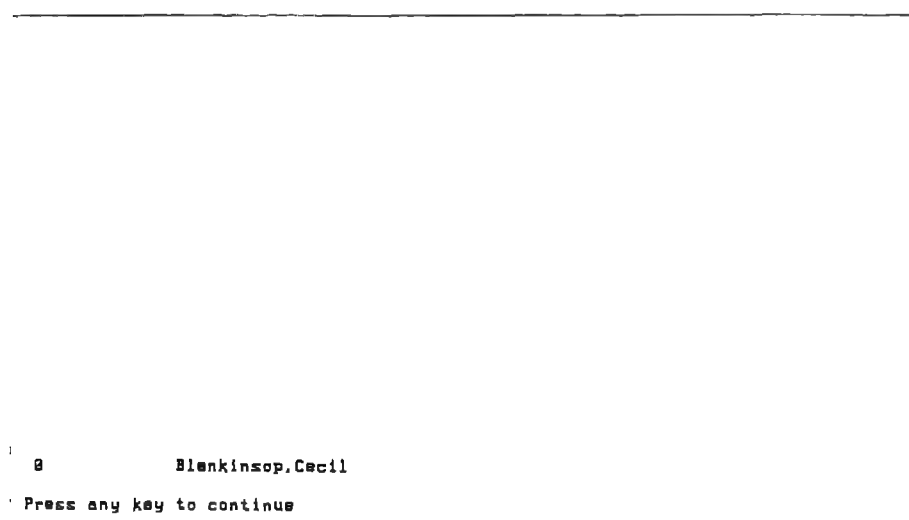
عملية تقليدية

توضح وتحدد قيماً لمتغيرات فى هذا القسم سواء كان ذلك مع ظهور أو مع عدم ظهور الفعل LET. ابدأ بتحميل بيسك السريع. (ارجع إلى البدء فى الدرس الثالث وفى الملحق B للمزيد من المعلومات عن بدء بيسك السريع).

١ - اكتب البرنامج التالى :



٢ - اضغط على Shift-F5 لتنفيذ البرنامج. تظهر المخرجات على النحو التالي :



لاحظ المخرجات، تم التحديد للمتغير FirstName\$ بدون عبارة LET، لا يستخدم المتغير Q في أى مكان باستثناء استخدامه في عبارة PRINT ويكون له قيمة تقليدية صحيحة تساوى صفراً، وهذه هي احدى المواقف التي يتم فيها انتاج متغير بدون توضيح رسمى.

٣ - اضغط على أى مفتاح للعودة إلى قائمة البرنامج.

٤ - انتقل إلى الدرس السادس والخمسين للاستمرار فى تسلسل التعلم.

الدرس السادس والسبعون

عبارة LINE

الوصف

ترسم عبارة LINE خطاً أو صندوقاً على الشاشة، وتكوينها هو كما يلي:

LINE STEP x1,y1 - STEP x2,y2,color,B|BF,style

يتسبب جزء STEP فى رسم الاحداثيات المعطاة بالنسبة إلى أحدث احداثيات مرسومة. الأجزاء x1, y1, x2, y2 هى الاحداثيات القطبية التى يرسم بينها الخط من x1, y1 إلى x2, y2. وجزء color الاختيارى هو اللون الذى يرسم به الخط، ويمكن استخدام الجزء B أو الجزء BF، ويحدد B أن المطلوب رسم صندوق أما BF فيحدد أن المطلوب هو ملء الصندوق. وفى أى من الحالتين يرسم صندوق بين محور الاحداثيات المعطاة، وجزء style هو غطاء من 16 بت يستخدم فى رسم نقاط الرسم على الشاشة، ويستخدم فى عمل شكل للخط line styling حيث تقرأ عبارة البت فى شكل معين من اليسار إلى اليمين، 1 بت ترسم نقاط اما 0 بت فلا تفعل ذلك. ويستخدم style مع الخطوط والصناديق المعتادة وليس له أى تأثير على الصناديق الملونة.

التطبيقات

تستخدم عبارة LINE إذا كانت امكانيات الرسومات متاحة فى رسم الخطوط. ويمكن أن تستخدم فى رسم خطوط الرسومات ورسومات الأعمدة (وذلك بخيار BF) ورسومات التقديم لتوضيح البيانات والأفكار.

عملية تقليدية

هذه العملية توضح استخدام عبارة LINE فى رسم رسومات أعمدة. يقبل البرنامج مدخلات عن بيانات الرسم ويرسم رسم أعمدة على الشاشة. استمر إذا كانت لديك امكانيات رسومات ملونة فقط. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```

File Edit View Search Run Debug Calls F1-Help
<Untitled>
' This program demonstrates the LINE statement.
DIM AX(10)

FOR Cnt = 1 TO 10
    READ t%
    AX(Cnt) = t%
NEXT

' Set the screen for medium resolution
SCREEN 1
CLS

PRINT "Your Graph: "
' Set the X axis
LINE (0, 180)-(280, 180)

' Set the starting position for the first graph

File Edit View Search Run Debug Calls F1-Help
<Untitled>
x1 = 0: y1 = 180: x2 = 20: y2 = 0

FOR Cnt = 1 TO 10
    t% = AX(Cnt)
    ' Compute the height of the bar
    y2 = t% * 2
    ' Draw the box
    LINE (x1 + 5, y1)-(x2, y2), Z, BF
    ' Move both x-coordinates to the right
    x1 = x1 + 20: x2 = x2 + 20
NEXT

DATA 13, 25, 44, 65, 32, 22, 8, 5, 71, 8

Immediate
Data: <Untitled> Context: Program not running

```

٢ - نفذ البرنامج. لاحظ استخدام عبارة LINE.

٣ - ارجع إلى البرنامج واحفظه كملف نصي تحت اسم LINE.BAS مع اخلاء الشاشة.

٤ - انتقل إلى الدرس الخامس والخمسين للاستمرار في تسلسل التعلم.

الدرس السابع والسبعون

عبارتا LINE INPUT و LINE INPUT#

الوصف

عبارة LINE INPUT : تقبل عبارة LINE INPUT مدخلات من لوحة المفاتيح، وتكوينها هو كما يلي :

```
LINE INPUT prompt string; string variable
```

جزء prompt string يشبه سلسلة الملقن في عبارة INPUT ويوضع بين علامتي تنصيص مزدوجتين، وجزء string variable هو مقصد البيانات التي يتم ادخالها، وتتوفر امكانيات التنقيح الكاملة مثل عبارة INPUT تماماً وتنتهي المدخلات مع عودة العربة، ولمعرفة التفاصيل عن رموز التنقيح ارجع إلى الدرس الثاني والستين، ولا تطبع عبارة LINE INPUT علامة استفهام (الا إذا كانت جزءاً من سلسلة الملقن). لاحظ أنه يوجد متغير واحد في عبارة LINE INPUT لقبول المدخلات، وتقرأ كل البيانات التي يتم ادخالها في هذا المتغير حتى تحدث حركة عودة العربة.

عبارة LINE INPUT# : تقرأ عبارة LINE INPUT# سطرأ من ملف تتابعي معين، وتكوينها هو كما يلي :

```
LINE INPUT #filenum, string var
```

جزء filenum هو رقم الملف المحدد له في عبارة OPEN، وجزء string var هو مقصد البيانات المقروءة، وتقرأ البيانات حتى تحدث حركة عودة العربة، وعبارة LINE INPUT# لها متغير واحد فقط للبيانات.

التطبيقات

عبارة LINE INPUT : عبارة LINE INPUT تكون مفيدة عندما يراد قراءة سلسلة من الرموز، وميزة عبارة LINE INPUT على عبارة INPUT هي أن الرموز المحددة تقرأ كذلك كجزء من البيانات، وفيما يلي بعض الأمثلة :

مثال ١

```
LINE INPUT "Enter text": TextIn$
```


يمكن أن تكون المدخلات كما يلي :

لاحظ في هذا المثال استخدام الفراغات والفاصلة. في عبارة INPUT يمكن للفاصلة أن تنهى عنصراً بيانات.

مثال ٢

```
LINE INPUT "?": MoreText$
```

مثال ٢

```
LINE INPUT: StillMore$
```

عبارة LINE INPUT# : تقارن عبارة LINE INPUT# مع عبارة LINE INPUT وتستخدم في قراءة بيانات محددة من ملف في متغير سلسلة. وفيما يلي بعض الأمثلة :

مثال ١

```
OPEN "Journal.Dat" FOR INPUT AS #2
...
LINE INPUT #2, JEntry$
PRINT JEntry$
```

مثال ٢

```
LINE INPUT #3, Line$
```

عملية تقليدية

هذه العملية توضح استخدام عبارات LINE INPUT و LINE INPUT#. ابدأ بتحميل ببسك السريع.

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls
<Untitled>
This program demonstrates the use of the LINE INPUT and LINE INPUT #
statements. The program accepts text, writes it to a file,
reads the text back and writes it to the screen.

CLS
PRINT "Enter text: (Enter blank line to terminate)"

OPEN "Notes.001" FOR OUTPUT AS #2

Line$ = " "
DO WHILE Line$ <> ""
    LINE INPUT ">": Line$
    WRITE #2, Line$
LOOP
CLOSE #2

CLS
PRINT "What you typed in is as follows: "

OPEN "Notes.001" FOR INPUT AS #2
DO WHILE NOT EOF(2)
    LINE INPUT #2, Line$
    PRINT Line$
LOOP

CLOSE #2

Immediate

Main: <Untitled> Context: Program not running 000016:001

```

٢ - نفذ البرنامج. لاحظ استخدام عبارات LINE INPUT و LINE INPUT# في البرنامج.
اكتب النص التالي. وعند انتهاء كل سطر اضغط على مفتاح الإدخال وعندما تصل إلى
نهاية الاختيار ادخل سطرًا فارغًا.

```

Enter text: (Enter blank line to terminate)
>THE PROMISE OF WORLD PEACE:
>A STATEMENT OF THE UNIVERSAL HOUSE OF JUSTICE TO THE PEOPLES OF THE WORLD.
>OCTOBER 1985.
>The Great Peace towards which people of goodwill throughout the centuries
>have inclined their hearts, of which seers and poets for countless generations
>have expressed their vision, and for which from age to age the sacred
>scriptures of mankind have constantly held the promise, is now at long last
>within the reach of the nations. For the first time in history it is possible
>for everyone to view the entire planet, with all its myriad diversified
>peoples, in one perspective. World peace is not only possible but
>inevitable. It is the next stage in the evolution of this planet. In the
>words of one great thinker, "the planetization of mankind."
>

```

٢ - بعد الضغط على مفتاح الإدخال مرة أخرى لإدخال سطرًا فارغًا تبين الشاشة ما يلي :

What you typed in is as follows:
THE PROMISE OF WORLD PEACE "
"A STATEMENT OF THE UNIVERSAL HOUSE OF JUSTICE TO THE PEOPLES OF THE WORLD."
OCTOBER 1985."
"The Great Peace towards which people of goodwill throughout the centuries"
"have inclined their hearts, of which seers and poets for countless generations"
"have expressed their vision, and for which from age to age the sacred"
"scriptures of mankind have constantly held the promise, is now at long last"
"within the reach of the nations. For the first time in history it is possible"
"for everyone to view the entire planet, with all its myriad diversified"
"peoples, in one perspective. World peace is not only possible but"
"inevitable. It is the next stage in the evolution of this planet, in the "
"words of one great thinker, 'the planetization of mankind.'"
""

Press any key to continue

٤ - اضغط على أى مفتاح للعودة إلى البرنامج. اختر New من قائمة File واكتب N لإخلاء الشاشة.

٥ - انتقل إلى الدرس المائة والثالث والستين للاستمرار فى تسلسل التعلم.

الدرس الثامن والسبعون

دالة LOC

الوصف

تعطى دالة LOC الوضع الحالى للملف محدد، وهى لها التكوين التالى :

LOC(filenum)

جزء filenum هو رقم الملف المحدد له فى عبارة OPEN. مع ملفات الاتصال العشوائى تعطى دالة LOC رقم آخر رمز تمت قراءته أو كتابته. ومع الملفات التتابعية تعطى دالة LOC موقع البايت الحالى مقسوماً على 128. ومع الملفات الثنائية تعطى دالة LOC موقع آخر بايت تمت قراءته أو كتابته. ومع وحدة COM تعطى دالة LOC عدد البايت الذى مازال موجوداً فى صف المدخلات. ولا يمكن استخدام دالة LOC مع الوحدات التالية : SCRN و KYBRD و LPTx حيث x رقم صحيح.

التطبيقات

تستخدم دالة LOC فى حفظ تتبع مكان الاتصال بالبيانات فى الملف.

وفيما يلى بعض الأمثلة :

مثال ١

```
OPEN "Temp.Dat" FOR RANDOM AS #2
..
IF LOC(2) = 0 THEN
..
ELSE
..
..
```

مثال ٢

```
OPEN "Test.Lst" FOR RANDOM AS #1
..
WHILE LOC(1) < 3000
...
WEND
```

عملية تقليدية

هذه العملية توضح استخدام دالة LOC. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :

```
File Edit View Search Run Debug Calls FI-Help
<Untitled>
This program demonstrates the use of the LOC function. The program reads
in a file and prints the data and the value returned by the LOC function.
The file read in was created in Module 74.

TYPE CustType
    CustName AS STRING * 25
    CustNum AS INTEGER
    CustType AS STRING * 2
END TYPE

DIM Customer AS CustType

OPEN "Cust.Fil" FOR RANDOM AS #2
RecCnt = 1
GET #2 1, Customer

DO WHILE NOT EOF(2)
    PRINT LOC(2): Customer.CustName, Customer.CustNum, Customer.CustType
    RecCnt = RecCnt + 1
    GET #2, RecCnt, Customer
LOOP

Immediate

Run: <Untitled> Context: Program not running 000.16:00
```

٢ - نفذ البرنامج. لاحظ استخدام دالة LOC في البرنامج. اضغط بعد ذلك على أي مفتاح للعودة إلى البرنامج.

```
1 Microsoft Inc.          333      A
2 More Money Corp.       1222     C
3 Singapore Scentis Co.  999      B

Press any key to continue
```

٣ - اختر New من قائمة File واختر عدم حفظ هذا البرنامج.

٤ - انتقل إلى الدرس المائة والخامس والعشرين للاستمرار في تسلسل التعلم.

الدرس التاسع والسبعون

عبارة LOCATE

الوصف

تستخدم عبارة LOCATE فى وضع نقطة البداية على الشاشة وفى التعريف الاختيارى لخواص الشاشة. وعادة ما تستخدم مع عبارات PRINT. وتكوين عبارة LOCATE هو كما يلى:

LOCATE row, column, cursor, start, stop

والخمس مكونات فى التكوين موصوفة فى الجدول التالى :

المؤشر	المعنى	النوع
Row	الصف الذى تظهر فيه نقطة البداية.	عدد صحيح
Column	العمود الذى تظهر فيه نقطة البداية.	عدد صحيح
Cursor	حالة نقطة البداية : (1) مرئية و (0) غير مرئية.	بوليان
Start	بداية سطر الفحص scan line لمعرفة موقع نقطة البداية.	عدد صحيح
Stop	نهاية سطر الفحص scan line لمعرفة موقع نقطة البداية.	عدد صحيح

وسطر الفحص scan line هو سطر يعرف حدود نقطة البداية. وفى حالة اللون الأحادى للشاشة يمكن أن تحتل نقطة البداية حتى 14 سطر فحص من 0 إلى 13. أما فى حالة النصوص الملونة فيمكنها أن تحتل 7 من 0 إلى 6. وفيما يلى بعض الأمثلة :

LOCATE 1,1

يضع هذا المثال نقطة البداية عند الركن العلوى الأيسر للشاشة.

LOCATE 1,1,0

يضع هذا المثال نقطة البداية مثل المثال السابق مع عدم رؤية نقطة البداية.

LOCATE 40,10,1,0,7

يضع هذا المثال نقطة البداية عند العمود 40 والسطر 10 ويجعل نقطة البداية مرئية وهي تحتل خلية الرمز كلها.

التطبيقات

تستخدم LOCATE في تحديد منطقة مخرجات على الشاشة واستخدام أمر PRINT مع LOCATE يكون قوياً ويعطيك تحكماً على موقع ظهور مخرجات البرنامج. انظر إلى عبارة LO-CATE في برنامج العينة الموجود في الدرس الثالث ولاحظ كيفية وضعها للصندوق وللنص داخل الصندوق.

عملية تقليدية

تكتب وتنفذ في هذه العملية برنامجاً باستخدام LOCATE. ابدأ بتحميل بيك السريع.

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls F1=Help
<Untitled>
CLS
Sel1$ = "Selection 1": Sel2$ = "Selection 2": Sel3$ = "Selection 3"
Sel4$ = "Quit": Sel5$ = "U -> UP / D -> DOWN / <CR> -> END"
LOCATE 2, 30: PRINT "MENU SELECTION"
LOCATE 4, 30: PRINT Sel1$
LOCATE 5, 30: PRINT Sel2$
LOCATE 6, 30: PRINT Sel3$
LOCATE 7, 30: PRINT Sel4$
LOCATE 9, 30: PRINT Sel5$

Immediate

Main: <Untitled> Context: Program not running 00010:001

```


٢ - اضغط على Shift-F5 لتنفيذ البرنامج. وتشبه المخرجات ما يلي:

```

MENU SELECTION
Selection 1
Selection 2
Selection 3
Quit
U -> UP / D -> DOWN / <CR> -> END

Press any key to continue
    
```

٣ - اضغط على أى مفتاح للعودة إلى قائمة البرنامج.

٤ - اضغط على Alt-F واضغط على مفتاح الإدخال واكتب N لإخلاء الشاشة دون أن تحفظ البرنامج.

٥ - انتقل إلى الدرس المائة وتسعة للاستمرار فى تسلسل التعلم.

الدرس الثمانون

عبارتا LOCK و UNLOCK

الوصف

تتحكم عبارتا LOCK و UNLOCK في الاتصال بملف مفتوح، وتكوينها هو كما يلي:

```
LOCK #filename, rec|start TO end
UNLOCK #filename, rec|start TO end
```

جزء #filename في كل من التكوينين هو رقم الملف المستخدم في فتح الملف في عبارة OPEN. جزء rec/start يحدد إما استخدام rec أو استخدام start. عندما تستخدم rec فيتم إغلاق أو عدم إغلاق السجل المطلوب، وعندما تستخدم start فإنها تحدد أول سجل أو بايت في مدى السجلات أو البايت المراد إغلاقه أو عدم إغلاقه. يحدد جزء end آخر بايت أو سجل يراد إغلاقه أو عدم إغلاقه.

وعندما يكون الملف مفتوحاً في الحالة الثنائية فتحدد أرقام السجلات مواقع البايت، وفي حالات أخرى، فإنها تشير إلى أرقام السجلات. وأقصى رقم للسجل يمكن استخدامه هو 2,147,483,647 وأقصى حجم للسجل هو 32,767 بايت.

التطبيقات

تستخدم عبارات LOCK و UNLOCK في بيئة الشبكة فقط لتقدم أو لتمنع الاتصال بالملف أو بجزء منه للعمليات الأخرى في البرنامج. وفيما يلي أمثلة لعبارات LOCK و UNLOCK:

مثال ١

```
OPEN "Interest" FOR RANDOM AS #2
LOCK #2, 100
UNLOCK #2
```

مثال ٢

```
OPEN "Acct.Pay" FOR INPUT AS #1
LOCK #1
UNLOCK #1
```

تفلق عبارة LOCK فى المثال الأول السجلات من 1 إلى 100 وتمنع عبارة UNLOCK اغلاق محتويات الملف. وفى المثال الثانى يكون الملف فى حالة تتابعية ولا تقدم عبارة LOCK مدى وذلك لأنه فى حالة الاتصال التتابعى تؤثر عبارة LOCK على الملف كله بغض النظر عن مدى السجلات المحدد.

عملية تقليدية

حيث إنه لا يمكن افتراض أنه متاح لك اتصال ببيئة شبكة كما أن مناقشة مثل هذه البيئة تقع خارج نطاق هذا الكتاب فلا يحتوى هذا القسم على مثال.

انتقل إلى الدرس المائة والتاسع والعشرين للاستمرار فى تسلسل التعلم.

الدرس الحادى والثمانون

دالة LOF

الوصف

تعطى دالة LOF حجم الملف بالبايت، وتكوينها هو كما يلى :

LOF(filename)

جزء filename هو رقم الملف المحدد للملف فى عبارة OPEN، ويعود حجم الملف بالبايت بغض النظر عن الحالة التى يكون مفتوحاً بها الملف. عندما تستخدم LOF مع عبارة OPEN COM فإن القيمة التى تعود هى عدد البايت الحر الموجود فى الذاكرة الاحتياطية للمخرجات، ولا يمكن استخدام دالة LOF مع الوحدات التالية : SCRIN أو KYBRD أو CONS: أو LPTx حيث x رقم صحيح.

التطبيقات

دالة LOF تكون مفيدة عندما تكون المعلومات الخاصة بحجم الملف ضرورية لتنفيذ البرنامج، تجنب الملفات الفارغة ويمثل عد عدد السجلات والتأكد من سعة القرص واختبار قيود البرنامج مواقع محددة من المواقع التى يمكن استغلال دالة LOF فيها، وفيما يلى بعض الأمثلة :

مثال ١

```
PRINT "Current file size is ":LOF(1)
```

مثال ٢

```
PRINT "Number of records : ":LOF(2)/RecordLen
```

مثال ٣

```
IF LOF(2) > 0 THEN
  ...
ELSE
  PRINT "Empty file. Cannot process."
END IF
```

عملية تقليدية

هذه العملية توضح استخدام دالة LOF، أبدأ بتحميل ببسك السريع.

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls F1-Help
<Untitled>
This program illustrates the LOF function. The program uses LOF to
give the file size in bytes.

TYPE CustType
    CustName AS STRING * 25
    CustNum AS INTEGER
    CustType AS STRING * 2
END TYPE

DIM Customer AS CustType
DIM

OPEN "Cust.Fil" FOR RANDOM AS #2
PRINT "File size in bytes:  LOF(2)"

CLOSE #2

----- Immediate -----
Run: <Untitled> Context: Program not running 00016:009

```

٢ - نفذ البرنامج. لاحظ استخدام دالة LOF في البرنامج.

```

File size in bytes: 285

Press any key to continue

```

٣ - اضغط على أى مفتاح للعودة إلى قائمة البرنامج.

٤ - اختر New واختار عدم حفظ البرنامج.

٥ - انتقل إلى الدرس الحادى والخمسين للاستمرار فى تسلسل التعلم.

الدرس الثانى والثمانون

عبارة LOG

الوصف

تعطى عبارة LOG اللوغاريتم الطبيعى لتعبير عددى معين، وتكوينها هى كما يلى :

`LOG(numeric expression)`

تعيد دالة LOG اللوغاريتم الطبيعى للأساس e (وهى حوالى 2.718282). وتحسب القيمة بدقة فردية كقيمة تقليدية. وعندما يكون التعبير العددي بدقة مزدوجة فتكون قيمة LOG المحسوبة فى دقة مزدوجة كذلك. ويقوم التعبير العددي بعدد أكبر من الصفر.

التطبيقات

تستخدم عبارة LOG عندما يكون مطلوباً حساب اللوغاريتم الطبيعى، وفيما يلى بعض الأمثلة:

```
PRINT LOG(n)
PRINT LOG(n) / LOG(10.0)
```

يوضح هذا المثال كيفية حساب اللوغاريتم للأساس 10 باستخدام دالة LOG.

عملية تقليدية

العملية التالية توضح استخدام دالة LOG، ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```

File Edit View Search Run Debug Calls
Untitled
This program computes logarithms to any positive base.
CLS
INPUT "Enter base and x ": b, x
Anylog = LOG(x) / LOG(b)
PRINT Log ", x: " to base ": b: " = ": Anylog

Immediate

Main: (Untitled) Context: Program not running 000000:001

```

٢ - نفذ البرنامج، اكتب 16 كأساس و 54 على أنها x ولاحظ استخدام دالة LOG في البرنامج.
اضغط على أى مفتاح لتعود إلى البرنامج.

```

Enter base and x ? 16,54
Log 54 to base 16 = 1.438722

Press any key to continue

```

٣ - من قائمة File اختر New واختر عدم حفظ البرنامج.
٤ - انتقل إلى الدرس المائة والحادى والثلاثين للاستمرار فى تسلسل التعلم.

الدرس الثالث والثمانون

دالة LPOS

الوصف

تعطى دالة LPOS الموقع الحالى لرأس الطابع داخل الذاكرة الاحتياطية للطابع. وتكونها
هو كما يلي :

LPOS(n)

جزء n هو رقم الطابع. وهو n الموجودة فى LPTn. ويختبر الطابع الموجود فى LPT1
باستخدام LPOS(1) كما يختبر الطابع الموجود فى LPT2 باستخدام LPOS(2) وهكذا.

التطبيقات

لا تعطى دالة LPOS الموقع الفعلى الواقعى لرأس الطابع. وإنما تعطى الموقع داخل الذاكرة
الاحتياطية للطابع وذلك لأن رموز الجداول لا تتسع داخل الذاكرة الاحتياطية للطابع. وفيما يلي
أمثلة لدالة LPOS.

مثال ١

```
LPRINT SPC(80 - LPOS(1)), "Captain"
```

مثال ٢

```
IF LPOS(1) > 60 THEN LPRINT
```

يستخدم المثال الثانى دالة LPOS فى الذهاب إلى سطر جديد على الطابع.

عملية تقليدية

هذه العملية تستخدم نفس البرنامج المستخدم فى الدرس الرابع والثمانين مع تعديله لطباعة
65 حرفاً فقط فى السطر. استمر فقط إذا ما كان لديك طابع متصل بالكمبيوتر. ابدأ بتحميل
بيسك السريع.

١ - حمل البرنامج المسمى LPRINT.BAS وغير فى البرنامج كما هو مبين فى القائمة التالية :


```

File Edit View Search Run Debug Calls F1=Help
LPRINT.BAS
DIM GUX(10)
CLS
PRINT "This program will print graph values below 63"

GETINPUT:
INPUT "ENTER NEXT VALUE FOR GRAPH, 0 TO END " I%
IF I% = 0 THEN GOTO DRAWGRAPH
GUX% = GUX% + 1: GUX(GUX%) = I%
IF (GUX% = 11) THEN GOTO GETINPUT
DRAWGRAPH:
CLS
LPRINT STRING$(10, "-"): "YOUR GRAPH": STRING$(10, "-"): PRINT : PRINT
FOR I = 1 TO 10:
  LPRINT : LPRINT I:
  FOR J = 1 TO GUX(I)
    IF LPOS(1) > 65 THEN
      LPRINT
      EXIT FOR
    ELSE LPRINT "X":
  END IF
NEXT
NEXT

```

Immediate

File: LPRINT.BAS Context: Program not running 00036:005

- ٢ - نفذ البرنامج ولاحظ استخدام دالة LPOS في البرنامج.
- ٣ - ارجع إلى البرنامج واحفظه كملف نصي تحت اسم LPOS.BAS مع اخلاء الشاشة.
- ٤ - انتقل إلى الدرس المائة والحادي والستين واستمر في تسلسل التعلم.

الدرس الرابع والثمانون

عبارتا LPRINT USING و LPRINT

الوصف

تطبع عبارتا LPRINT و LPRINT USING بيانات بواسطة الطابعة والعبارتان متشابهتان جداً مع عبارتي PRINT و PRINT USING فيما عدى أن المخرجات تتجه إلى الطابع بدلاً من اتجاهها للشاشة. وتكوينهما هو كما يلي :

```
LPRINT expression list
LPRINT USING format string: expression list
```

جزء expression list فى كل من التكوينين هو قائمة بالمتغيرات والثوابت والتعبيرات المراد طباعتها. وجزء format string يصف شكل طباعة البيانات. وللوصف الكامل لقائمة المتغيرات وسلسلة الشكل ارجع إلى الدرسين المائة وتسعة والمائة وعشرة. وتتجه المخرجات إلى الطابع الموجود فى البوابة LPT1.

التطبيقات

تستخدم عبارتا LPRINT و LPRINT USING فى طباعة بيانات على الطابع. وتفترض العبارتان أن عرض سطر الطابع 80 خانة. وفيما يلى أمثلة لعبارات LPRINT و LPRINT USING :

مثال ١

```
FmtStr$ = "/ /"
LPRINT USING FmtStr$: "Money"; "Money"; "And"; "More Money"
```

مثال ٢

```
LPRINT FNLFmt$(NewStr$),NewStr$
```

مثال ٣

```
FOR Cnt = 1 TO 12
    LPRINT USING "!": Q(Cnt)
NEXT
```

عملية تقليدية

هذه العملية تستخدم البرنامج المعد في الدرس المائة والثالث والأربعين وتطبع المخرجات على طابع. استمر إذا ما كان لديك طابع متصل بجهاز الكمبيوتر فقط. ابدأ بتحميل بيسك السريع.

١ - حمل البرنامج STRING. BAS وعدل عبارة PRINT لـ هو مبين في القائمة التالية :

```
File Edit View Search Run Debug Calls F1-Help
STRING.BAS
DIM G%(10)
GETINPUT:
INPUT "ENTER NEXT VALUE FOR GRAPH, 0 TO END ", I%
IF I% = 0 THEN GOTO DRAWGRAPH
GUS% = GUS% + 1: GUX%(GUS%) = I%
IF GUS% < 11 THEN GOTO GETINPUT
DRAWGRAPH:
CLS
LPRINT STRING$(10, "-"): "YOUR GRAPH": STRING$(10, "-"): PRINT : PRINT
FOR I = 1 TO 10: LPRINT I: STRING$(GUX%(I), "X"): NEXT
```

Immediate

Name: STRING.BAS Context: Program not running C 00010:019

٢ - نفذ البرنامج ولاحظ استخدام عبارة LPRINT.

٣ - اكتب 10 واضغط على مفتاح الإدخال. اكتب 5 و 8 و 13 و 0 مع الضغط على مفتاح الإدخال في كل مرة يتم كتابة رقم من هذه الأرقام.

```
ENTER NEXT VALUE FOR GRAPH, 0 TO END 10
ENTER NEXT VALUE FOR GRAPH, 0 TO END 5
ENTER NEXT VALUE FOR GRAPH, 0 TO END 8
ENTER NEXT VALUE FOR GRAPH, 0 TO END 11
ENTER NEXT VALUE FOR GRAPH, 0 TO END 3
ENTER NEXT VALUE FOR GRAPH, 0 TO END 8
```

٤ - يجب أن تشبه مخرجاتك ما يلي :

```

.
-----YOUR GRAPH-----
1 *****
2 *****
3 *****
4 *****
5 ***
6
7
8
9
10

```

٥ - ارجع إلى البرنامج واحفظ هذا البرنامج كملف نصي تحت اسم LPRINT. BAS مع اخلاء الشاشة.

٦ - انتقل إلى الدرس الثالث والثمانين للاستمرار في تسلسل التعلم.

الدرس الخامس والثمانون

عبارتا LSET و RSET

الوصف

تنقل عبارتا LSET و RSET بيانات إلى ذاكرة الملف الاحتياطية من الذاكرة أو تضبط بيانات سلسلة من اليسار أو من اليمين على التوالي في متغير سلسلة. وتكوينها هو ما يلي :

تكوين عبارة LSET :

```
LSET string var = string expression
```

جزء string var هو حقل معرف في عبارة FIELD أو متغير سلسلة. وجزء string ex-pression هو قيمة محددة لمتغير السلسلة. وعندما يكون متغير السلسلة أكبر من تعبير السلسلة فتضبط البيانات من الناحية اليسرى مع تكملتها بفراغات من الناحية اليمين. وعندما يكون المتغير أقل من تعبير السلسلة فتتلفى الخانات الزائدة، ويجب أن تتحول القيم العددية إلى تعبيرات سلاسل باستخدام MKI\$ أو MKD\$ أو MKL\$ أو MKS\$ قبل استخدام عبارة LSET لنقل البيانات إلى ذاكرة الملف الاحتياطية أو إلى الضبط من ناحية اليسار للسلسلة.

تكوين عبارة RSET :

```
RSET string var = string expression
```

جزء string var و string expression متطابقان تماماً مع ما هو موجود في عبارة LSET. ويتم تشغيل البيانات بنفس الطريقة مثل عبارة LSET باستثناء أن التضبط يحدث من الناحية اليمنى بدلاً من حدوثه من الناحية اليسرى. وجميع القيود التي تقع على LSET تقع كذلك على RSET.

التطبيقات

تستخدم عبارتا LSET و RSET في اعداد بيانات للكتابة في ملف اتصال عشوائى. وفيما يلي بعض الأمثلة :

عبارة LSET :

```
Size$ = SPACE$(25)
FirstName$ = "Malcolm"
LSET Size$ = FirstName$
```

يبين هذا المثال كيفية استخدام عبارة LSET في تشكيل المتغير FirstName\$ في سلسلة طولها 25 خانة مضبوطة من ناحية اليسار. من الممكن كذلك تحديد متغير سجل لمتغير سجل آخر بون الحذف من توافقية نوع السجل. ويحدث ذلك على النحو التالي :

```
TYPE t1
  i1 A INTEGER
  i2 AS STRING * 10
END TYPE

TYPE t2
  s1 AS STRING * 20
END TYPE

DIM r1 AS t1, r2 AS t2
LSET r1 = r2
```

ينسخ في هذا المثال متغير السجل r2 في متغير السجل r1.
عبارة REST :

```
LS$ = SPACE$(12)
Cuisine$ = "Mexican"
RSET LS$ = Cuisine$
```

يبين هذا المثال كيفية استخدام RSET في تشكيل المتغير Cuisine\$ في سلسلة طولها 12 خانة مضبوطة من ناحية اليمين. وفيما يلي مثال لتحديد متغير سجل لمتغير سجل آخر له نوع مختلف عن نوعه باستخدام عبارة RSET :

```
TYPE t1
  Sample1 AS STRING * 10
END TYPE

TYPE t2
  Sample2 AS STRING * 20
END TYPE

DIM Rec1 AS t1, Rec2 AS t2
RSET Rec1 = Rec2
```

عملية تقليدية

هذه العملية توضح استخدام عبارتي LSET و RSET. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Exit
(Untitled)
'This is a demo for the LSET and RSET statements.
'The program accepts input, formats the input and prints it on the screen.

CLS
StudentName$ = SPACES(25): Score$ = SPACES(5)
PRINT "Student name": TAB(40): "Student Score": PRINT STRING$(60, "-")
FOR cnt = 1 TO 10
  LOCATE cnt + 2, 1: INPUT SName$
  LOCATE cnt + 2, 40: INPUT SScore$
  LSET StudentName$ = SName$
  LOCATE cnt + 2, 1: PRINT StudentName$:
  RSET Score$ = SScore$
  LOCATE cnt + 2, 40: PRINT Score$

NEXT
END

```

Immediate

Main: (Untitled) Context: Program not running 00017:004

٢ - نفذ البرنامج. لاحظ استخدام عبارات LSET و RSET في نقل بيانات إلى الذاكرة الاحتياطية قبل طباعتها في البرنامج. اكتب البيانات التالية مع الضغط على مفتاح الإدخال بعد كل عملية إدخال. ومع اتمامك لكل سطر يعاد تشكيل البيانات. اضغط على Ctrl-Break لايقاف البرنامج.

Student name	Student Score
Mary Jane	34
Simone A.	55
June Ally	123
?	

٣ - اضغط على أى مفتاح للعودة إلى البرنامج. اختر New من قائمة File واكتب N لإخلاء الشاشة.

٤ - انتقل إلى الدرس المائة والحادى عشر للاستمرار فى تسلسل التعلم.

الدرس السادس والثمانون

دالتا LTRIM\$ و RTRIM\$

الوصف

تزيل دالتا LTRIM\$ و RTRIM\$ الفراغات السابقة أو التالية للبيانات على التوالي وذلك من تعبير السلسلة. وتكوينها هو كما يلي :

```
LTRIM$(string exp)
RTRIM$(string exp)
```

جزء string exp فى كل من التكوينين يمثل أى تعبير سلسلة، ويمكن أن يكون تعبير السلسلة متغير سلسلة أو ثابت سلسلة أو تعبيراً ينتج عنه سلسلة. وتقبل كل من الدالتين سلاسل ثابتة الطول أو متغيرة الطول كمؤشرات لها.

التطبيقات

تستخدم دالتا LTRIM\$ و RTRIM\$ فى تشكيل السلاسل للطباعة أو لأى غرض آخر فى معالجة بيانات السلاسل، وفيما يلي بعض الأمثلة :

مثال ١

```
Tst$ = "Please wait..  "
PRINT RTRIM$(Tst$)
```

مثال ٢

```
DIM St2 AS STRING * 35
St2 = "NOT the RED button!"
PRINT LTRIM$(RTRIM$(St2))
```

عملية تقليدية

تقوم فى هذه العملية بتعديل البرنامج المقدم فى الدرس الثانى والسبعين. وهو ULCASE. BAS، لتوضيح نوال LTRIM\$ و RTRIM\$. ابدأ بتحميل بيسك السريع.

١ - من قائمة File اختر Open واضغط على Tab للذهاب إلى الدليل. اختر ULCASE.BAS واضغط على مفتاح الإدخال.

٢ - اضغط على Shift-F2 لتنقيح الدالة GetChar\$.

٣ - لاحظ البرنامج التالي والدالة الموصوفة. استمر سطرًا بسطر خلال وصفك للدالة وعدل الدالة لتمثل القائمة التالية :

```

File Edit View Search Run Debug Calls F1=Help
ULCASE.BAS
' This program demonstrates the use of LTRIMS and RTRIMS functions.
' This program is modified from Module 75.

DECLARE FUNCTION GCh$(UCH AS STRING, x AS INTEGER, y AS INTEGER, P AS STRING)
DIM UCh AS STRING * 30, P AS STRING * 75
CLS
LOCATE 23, 1
PRINT "E)dit / C)reate / D)elete / Q)uit ?"
UCh = "EDCQ"
P = "Enter Selection"
Selection$ = GCh$("EDCQ", 24, 1, P)

Immediate

Main: ULCASE.BAS Context: Program not running 00010:002

```

٤ - اضغط على Shift-F2 لتنقيح الجزء الرئيسي. يمكن حذف الأسطر بالضغط على Ctrl-Y.

احذف التعليقات الموجودة في قمة البرنامج وأضف التعليقات الجديدة كما هو مبين في القائمة التالية. أضف عبارة DIM كما هو موضح. عدل بقية البرنامج كما هو موضح كذلك.

```

File Edit View Search Run Debug Calls F1=Help
ULCASE.BAS
' This program demonstrates the use of LTRIMS and RTRIMS functions.
' This program is modified from Module 75.

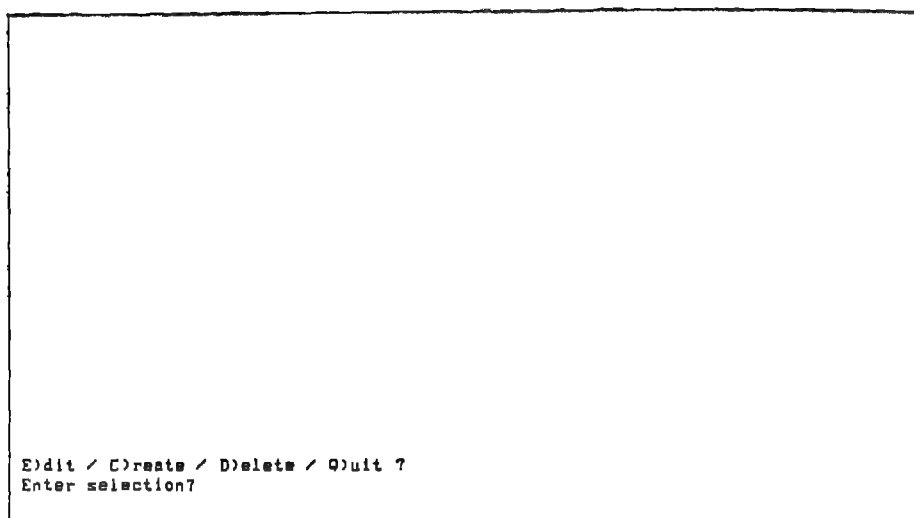
DECLARE FUNCTION GCh$(UCH AS STRING, x AS INTEGER, y AS INTEGER, P AS STRING)
DIM UCh AS STRING * 30, P AS STRING * 75
CLS
LOCATE 23, 1
PRINT "E)dit / C)reate / D)elete / Q)uit ?"
UCh = "EDCQ"
P = "Enter Selection"
Selection$ = GCh$("EDCQ", 24, 1, P)

Immediate

Main: ULCASE.BAS Context: Program not running 00011:003

```

٥ - نفذ البرنامج ولاحظ استخدام دالة LTRIM\$ ودالة RTRIM\$. تظهر مدخلاتك على نفس سطر الملقن. تحل المدخلات الجديدة محل المدخلات القديمة. وعندما يتم ادخال مدخلات مناسبة فتستبدل المدخلات بالاختيار الذي تجريه.



- ٦ - اضغط على أى مفتاح للعودة إلى البرنامج.
- ٧ - من قائمة File اختر Save As واحذف اسم الملف ULCASE.BAS بالضغط على قضيب المسافات. اكتب LTRIM.BAS كاسم ملف جديد واضغط على مفتاح الادخال.
- ٨ - من قائمة File اختر New مع اخلاء الشاشة.
- ٩ - انتقل إلى الدرس الخامس والستين للاستمرار فى تسلسل التعلم.

الدرس السابع والثمانون

دالة وعبارة MID\$

الوصف

تعمل الكلمة المحجوزة MID\$ بطريقتين طبقاً لموقع ظهورها بالنسبة إلى عبارة التحديد. فتعمل MID\$ كدالة إذا ما ظهرت في الطرف الأيمن لعبارة التحديد وتعيد جزءاً محدداً من سلسلة المؤشر. كما تعمل MID\$ كعبارة إذا ما ظهرت في الطرف الأيسر لعبارة التحديد وتستبدل جزءاً محدداً من سلسلة المؤشر بتعبير سلسلة جديد. والكلمة المحجوزة MID\$ هي إحدى كلمات البيسك الأكثر تعداداً للاستخدام. وتكوينها هو كما يلي :

التكوين الأول (كدالة) :

MID\$(String expression, start, length)

الوصف	الجزء
كلمة من كلمات بييسك المحجوزة، وتعيد في هذا التكوين الجزء المحدد من تعبير السلسلة.	MID\$
سلسلة مؤشر يستخلص لها جزء مع اعادته.	string expression
موقع البداية للاستخلاص من تعبير السلسلة. فإذا كان موقع البداية أكبر من طول السلسلة فتعيد MID\$ سلسلة فارغة.	start
مؤشر اختياري يحدد عدد الرموز المراد استخلاصها. ويجب أن يقع بين 1 و 32,767. فإذا لم يتحدد أو إذا كانت هناك رموز أقل في السلسلة عن الطول من البداية فتعيد MID\$ كل الرموز من البداية.	length

التكوين الثاني (كعبارة) :

MID\$(String variable, start, length) = String expression

الجزء	الوصف
MID\$	كلمة من كلمات بيسك المحجوزة.
string variable	مقصد التحديد. ولا يسمح إلا بمتغيرات.
start	موقع بداية الاستبدال في سلسلة المقصد.
length	مؤشر اختياري يحدد عدد الرموز المراد استبدالها. فإذا لم يكن محدداً فيستخدم تعبير السلسلة كله.
string expression	سلسلة المصدر التي يعدها التحديد لمتغير سلسلة المقصد. ويمكن أن تكون متغيراً أو ثابتاً أو تعبيراً.

التطبيقات

يمكن أن تستخدم الكلمة المحجوزة MID\$ في تطبيقات معالجة سلاسل واسعة النطاق. واستخلاص جزء من سلسلة واستبدال جزء من سلسلة من أهم الاستخدامات الشائعة لها. وفيما يلي بعض الأمثلة :

```
Prompt$ = "G Men! Cheese it."
Prompt2$ = "Fuzz !"
PRINT Prompt$
MID$(Prompt$,1,6) = Prompt2$
PRINT Prompt$
```

المخرجات

```
G Men! Cheese it.
Fuzz ! Cheese it.
```

يوضح هذا المثال استخدام عبارة MID\$. تستبدل السلسلة الجزئية "G MEN!" بـ Pro-
mpt 2\$.

```
Prompt$ = "Here he goes again.": Prompt2$ = "What "
Prompt2$ = Prompt2$ + MID$(Prompt$,14,5) + "7"
PRINT Prompt$
PRINT Prompt2$
```

المخرجات

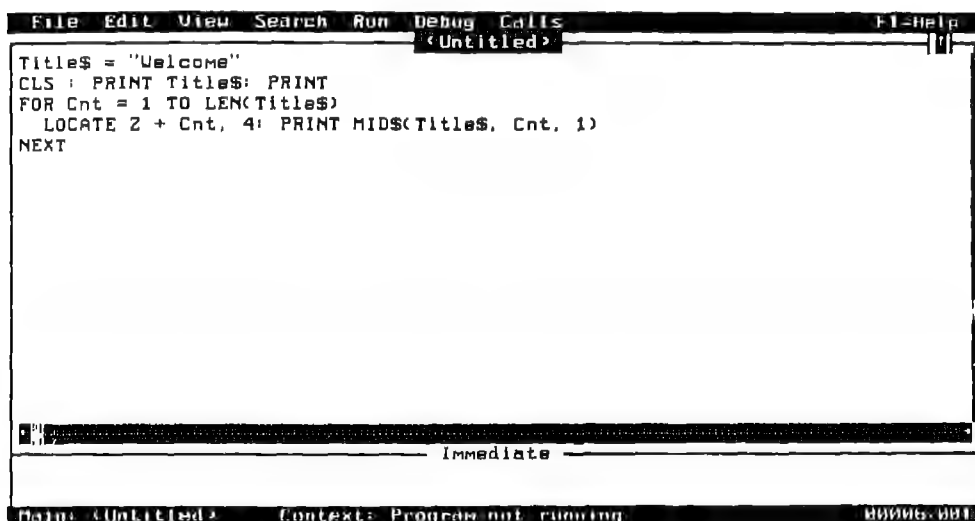
Here he goes again.
What again?

يوضح هذا المثال استخدام MID\$ كدالة تعيد سلسلة جزئية من تعبير سلسلة.

عملية تقليدية

عادة ما يطبع النص على الشاشة أفقياً. وهناك طريقة مختلفة وفريدة عن هذه الطريقة وهي العرض رأسياً. هذه العملية تستخدم MID\$ في برنامج لطباعة سلسلة أفقياً ورأسياً. ابدأ بتحميل ببسك السريع.

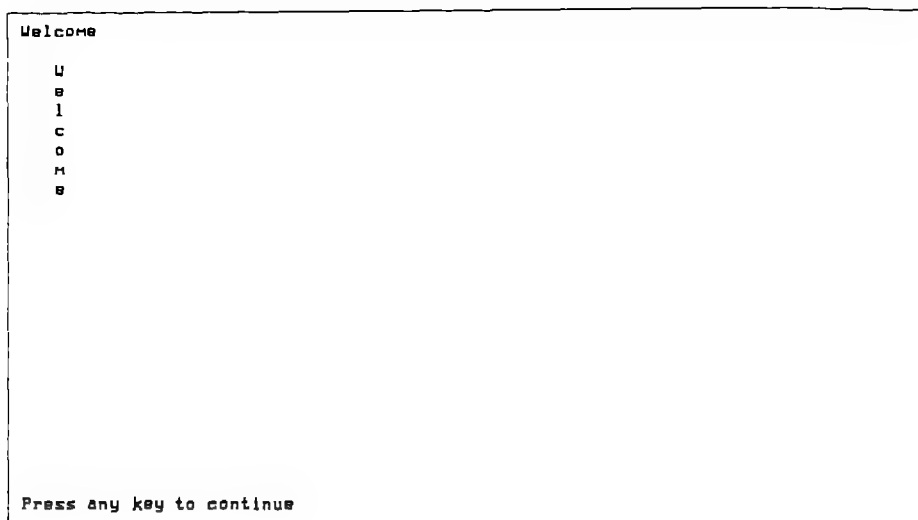
١ - اكتب البرنامج التالي :



```

File Edit View Search Run Debug Calls F1=Help
<Untitled>
Title$ = "Welcome"
CLS : PRINT Title$: PRINT
FOR Cnt = 1 TO LEN(Title$)
    LOCATE 2 + Cnt, 4: PRINT MID$(Title$, Cnt, 1)
NEXT
Immediate
Main: <Untitled> Context: Program not running 000000.0001
    
```

٢ - نفذ البرنامج. لاحظ المخرجات على الشاشة واستخدام MID\$ في تحقيق الطباعة الرأسية.



٣ - اضغط على أى مفتاح للعودة إلى البرنامج، اضغط على Alt-F ثم اضغط على مفتاح الإدخال واكتب N لإخلاء الشاشة.

٤ - انتقل إلى الدرس المائة والعشرين للاستمرار فى تسلسل التعلم.

الدرس الثامن والثمانون

دوال MKD\$ و MKI\$ و MKL\$ و MKS\$

الوصف

تقوم الدوال MKD\$ و MKI\$ و MKL\$ و MKS\$ بتحويل قيم عددية إلى قيم سلاسل مناظرة لها. وتكوينها هو كما يلي :

```
MKI$(integer exp)
MKS$(single precision exp)
MKL$(long integer exp)
MKD$(double precision exp)
```

تحويل دالة MKI\$ تعبير عددي صحيح إلى سلسلة من 2 البايت.

تحويل دالة MKS\$ تعبير له دقة فردية إلى سلسلة من 4 بايت.

تحويل دالة MKL\$ تعبير عددي صحيح طويل إلى سلسلة من 4 بايت.

تحويل دالة MKD\$ تعبير له دقة مزدوجة إلى سلسلة من 8 بايت.

تستخدم هذه الدوال مع عبارات FIELD و PUT في كتابة اعداد في ملف وتحويل الدوال القيم العددية إلى سلاسل بحيث يمكن تخزينها في سلاسل معرفة في عبارة FIELD.

التطبيقات

دوال MKD\$ و MKI\$ و MKL\$ و MKS\$ مفيدة في تحويل القيم العددية إلى سلاسل قبل تخزينها في ملف. وفيما يلي مثال لذلك :

```
OPEN "SalesTx.Dat" FOR RANDOM AS #3
..
FIELD #3 20 AS ItemName$, 10 AS Qty$, 12 AS SalesTx$
..
INPUT "Enter Item ";ItemName$
INPUT "Quantity ";Qty$
INPUT "Sales tax ";ST!
Qty$ = MKI$(Qty): SalesTx$ = MKS$(ST!)
..
PRINT #3, ItemName$, Qty$, SalesTx$
```


يوضح المثال كيفية استخدام دالة MKI\$ ودالة MKS\$ في تحويل قيم عددية صحيحة وأخرى ذات دقة فردية إلى سلاسل بحيث يمكن كتابتها في الملف SalesTx. Dat. ويمكن أن يتحقق نفس التأثير باستخدام أنواع سجلات وملفات مرتبة لأنواع سجلات من هذه الأنواع. ويحدث ذلك عن طريق توضيح نوع سجل يعرفه المستفيد بعبارات TYPE و END TYPE وتعريف ملف يحتوى على سجلات من هذا النوع. ويوضح المثال التالى نفس العملية باستخدام عبارات TYPE و END TYPE. لاحظ بساطة الشفرة بالمقارنة بالمثال السابق.

```

TYPE SalesTx
  ItemName AS STRING * 20
  Qty      AS STRING * 10
  STax     AS STRING * 12
END TYPE

DIM STRec AS SalesTx
OPEN "SalesTx.Dat" FOR RANDOM AS #3 LEN = LEN(STRec)

INPUT "Enter item ";STRec.ItemName
INPUT "Quantity   ";STRec.Qty
INPUT "Sales tax  ";STRec.STax

PRINT #3,STRec

```

عملية تقليدية

هذه العملية توضح استخدام بوال MKD\$ و MKI\$ و MKS\$. ابدأ بتحميل ببسك السريع.

١ - اكتب البرنامج التالى :

```

File Edit View Search Run Debug Calls File Help
Untitled
This program illustrates the use of MKS$, MKD$, MKI$, and MKL$ functions.
The program creates a file using these functions.

OPEN "Cust2.Fil" FOR RANDOM AS #2
FIELD #2, 25 AS CName$, 2 AS CNum$, 8 AS CreditLim$, 4 AS LastInv$

DO WHILE UCASE$(Choice$) <> "Y"
  INPUT "Enter customer name: "; Cn$
  INPUT "Customer number   : "; Cnumber

```

```

INPUT "Credit limit      : "; CrLim#
INPUT "Last invoice     : "; LInu#
INPUT "Done ? (Y/N)    "; Choices$
LSET CName$ = Cn$
LSET CNum$ = MKI$(Cnumber)
LSET CreditLim$ = MKD$(CrLim#)
LSET LastInu$ = MKS$(LInu!)
PUT #2
LOOP
CLOSE #2
RecCnt = 1

```

Immediate

Main: "Untitled" Context: Program not running

000 16:020

٢ - نفذ البرنامج. لاحظ استخدام نوال MKD\$ و MKI\$ و MKS\$ في البرنامج. اكتب البيانات التالية مع الضغط على مفتاح الادخال بعد كل عملية ادخال.

```

Enter customer name: 7 Mission Impossible Inc.
Customer number    : 7 1233
Credit limit      : 7 300000.00
Last invoice      : 7 12000
Done ? (Y/N)      ? N
Enter customer name: 7 Last Resort Motel
Customer number    : 7 888
Credit limit      : 7 25000.00
Last invoice      : 7 3000
Done ? (Y/N)      ? n
Enter customer name: 7 Income Only Corp.
Customer number    : 7 3433
Credit limit      : 7 1000000
Last invoice      : 7 120000
Done ? (Y/N)      ? Y

```

Press any key to continue

٣ - ارجع إلى البرنامج واختر New دون أن تحفظ البرنامج.

٤ - انتقل إلى الدرس السادس والعشرين للاستمرار في تسلسل التعلم.

الدرس التاسع والثمانون

دوال MKDMBF\$ و MKSMBF\$ و CVDMBF و CVSMBF

الوصف

تتعامل هذه الدوال بصفة خاصة مع أشكال ميكروسوفت الثنائية Microsoft Binary For- mat وهي طريقة لتمثيل الأعداد الحقيقية داخلياً، (لاحظ جزء MBF من الدوال)، تحول هذه الدوال الأعداد المخزنة في MBF (أشكال ميكروسوفت الثنائية) إلى أشكال IEEE (معهد المهندسين الكهربائيين والالكترونيين Institute of Electric and Electronic Engineers) والعكس، وهذا مفيد بصفة خاصة حيث إن الصيغ القديمة لبيسك ميكروسوفت تستخدم صيغة MBF في التمثيل الداخلي وتسمح لك هذه الدوال باستخدام ملفات اتصال عشوائى سبق انتاجها بواسطة صيغ ببسك القديمة، وتقدم اشكال IEEE بعض مزايا اضافية عن أشكال MBF فى النواحي التالية :

- تشكيل IEEE له رقم أو رقمان اضافيان فى الجزء العشري يعطى مدى أوسع للأس.

- تشكيل IEEE له المدى التالى :

النوع	المدى
single precision	من صفر 3.37×10^{-38} إلى -8.43×10^{-37} من صفر 8.43×10^{-37} إلى 3.37×10^{-38}
double precision	من -1.67×10^{308} إلى -4.19×10^{-307} من صفر 4.19×10^{-307} إلى -4.19×10^{308}

تصل دقة تشكيلات الأعداد فردية الدقة إلى حوالى 7 أرقام عشرية وتصل دقة الأعداد مزدوجة الدقة إلى من 15 إلى 16 رقماً عشرياً فى تشكيل IEEE.

ويتم تحويل البرامج القديمة إلى بيسك السريع بطريقتين :

- في ترجمة منفصلة باستخدام خيار ./mbf.

- بتعديل ملفات بيانات الاتصال العشوائى من البرامج القديمة وإعادة ترجمة البرنامج.

وفيما يلى تكوين نوال MKDMBF\$ و MKSMBF\$ و CVDMBF و CVSMBF.

```
MKDMBF$(double-precision number)
MKSMBF$(single-precision number)
CVDMBF$(eight-byte string)
CVSMBF$(four-byte string)
```

وفيما يلى جدول بأوصاف النوال :

الوصف	الدالة
تحويل أعداداً لها دقة مزدوجة إلى سلسلة من 8 بايت	MKDMBF\$
تحويل أعداداً لها دقة فردية إلى سلسلة من 4 بايت.	MKSMBF\$
تحويل سلسلة من 8 بايت إلى عدد له دقة مزدوجة.	CVDMBF
تحويل سلسلة من 4 بايت إلى عدد له دقة فردية.	CVSMBF

التطبيقات

الاستخدام الأولى للنوال MKDMBF\$ و MKSMBF\$ و CVDMBF و CVSMBF هو

جعلك قادراً على استخدام ملفات اتصال عشوائى سبق انتاجها باستخدام صيغ بيسك قديمة من شركة ميكروسوفت. وفيما يلى مثال لذلك :

```
TYPE OldDat
  OD1 AS STRING * 20
  OD2 AS STRING * 4
END TYPE
DIM ODRec AS OldDat
OPEN "Account.Old" FOR RANDOM AS #2 LEN = LEN(ODRec)
...
GET #2
OD2Num! = CVSMBF(OD2)
```

عملية تقليدية

هذه العملية تصف بوال MKDMBF\$ و MKSMBF\$ و CVDMBF و CVSMBF.

ابدأ بتحميل ببسك السريع.

١ - اكتب البرنامج التالي :

```
File Edit View Search Run Debug Exit Help
Uncompiled
This program demonstrates the use of MKSMBF$, MKDMBF$, CUSMBF and
CUDMBF functions. The program creates a file and reads it back.

OPEN 'Cust3.Fil' FOR RANDOM AS #2
FIELD #2, 25 AS CNames, 8 AS CreditLim, 4 AS LastInv
CLS

DO WHILE LCASE$(Choice$) <> "Y"
  INPUT "Enter customer name: "; Cn$
  INPUT "Credit limit: "; CrLim$
  INPUT "Last invoice: "; Lin$
  INPUT "Done ? (Y/N) "; Choice$
  LSET CNames = Cn$
  LSET CreditLim = MKDMBF$(CrLim$)
  LSET LastInv = MKSMBF$(Lin$)
  PUT #2
LOOP

CLOSE #2

OPEN 'Cust3.Fil' FOR RANDOM AS #2
FIELD #2, 25 AS CNames, 8 AS CreditLim, 4 AS LastInv
GET #2, 1

DO WHILE NOT EOF(2)
  PRINT CNames: CUDMBF$(CreditLim): CUSMBF$(LastInv)
  GET #2
LOOP

CLOSE #2

----- Immediate -----
Enter Uncompiled: Command: Program not running 00000000
```

٢ - نفذ البرنامج ولاحظ استخدام بوال MKDMBF\$ و MKSMBF\$ و CVDMBF و CVSMBF. ادخل البيانات التالية :

Enter customer name:	7 Moonlighting Co.
Credit limit	7 200000
Last invoice	7 12000
Done ? (Y/N) ? n	
Enter customer name:	7 Precocious Kids Inc.
Credit limit	7 200000
Last invoice	7 12000
Done ? (Y/N) ? y	
Moonlighting Co.	200000 12000
Precocious Kids Inc.	200000 12000

Press any key to continue

٣ - ارجع إلى البرنامج واختر New نون أن تحفظ هذا البرنامج.

٤ - انتقل إلى الدرس المائة والتاسع عشر للاستمرار في تسلسل التعلم.

الدرس التسعون

عبارة NAME.. AS..

الوصف

تستخدم عبارة NAME..AS.. فى اعادة تسمية ملف على قرص، وتكوينها هو كما يلى :

NAME old name AS new name

المؤشران old name و new name عبارة عن أسماء ملفات صحيحة من DOS مع مواصفات مسار اختيارية. ويجب أن توضع الأسماء بين علامتى تنصيص. وعلى عكس أمر RENAME لاعادة التسمية من نظام DOS فيمكن استخدام عبارة NAME..AS.. فى نقل ملف من أحد الأدلة إلى دليل آخر ولكن على نفس القرص، ولا يمكن اعادة تسمية الدلائل باستخدام NAME.. AS.. .

التطبيقات

تستخدم عبارة NAME..AS.. فى اعادة تسمية ملفات اثناء تنفيذ البرنامج، وفيما يلى بعض الأمثلة :

```
NAME "PCOMM.CFG" AS "PCOMM.BAK"
NAME "\\GAMES\\PIKMAN.EXE" AS "\\GAMES\\PIKMAN.OLD"
NAME "\\UTIL\\PCFORMAT.COM" AS "\\DOS\\PCFORMAT.COM"
```

يوضح المثال الأخير نقل ملف عبر دلائل باستخدام عبارة NAME.. AS...

عملية تقليدية

العملية التالية توضح استخدام عبارة NAME.. AS.. ابدأ بتحميل ببسك السريع.

١ - اكتب البرنامج التالى :

```

FILE Edit View Search Run Debug Calls F1=Help
Untitled
This program demonstrates the NAME .. AS .. statement
CLS
FILES "M.BAS"
INPUT "Enter original filename (Enter to end)": F1$
IF F1$ <> "" THEN
    INPUT "Enter new filename (Enter to end)": F2$
    IF F2$ <> "" THEN NAME F1$ AS F2$
    ELSE PRINT "Thank you for participating !": END
ELSE
    PRINT "Thank you for participating !": END
END IF
INPUT "Press Enter to continue": C$
PRINT
PRINT "Following is a list of the current filenames to prove we did it."
FILES "M.BAS"
Immediate
Name: Untitled Contents: Program not running 0001:014

```

٢ - نفذ البرنامج. لاحظ استخدام NAME.. AS.. في البرنامج. اضغط على مفتاح الإدخال لإنهاء البرنامج دون إجراء أى تغيير على أسماء الملفات.

```

C:\QB
SAMPLE .BAS    REMLINE .BAS    SORTDEMO.BAS    TORUS .BAS
DEMO1 .BAS     DEMO2 .BAS     DEMO3 .BAS     INCH2CM .BAS
BOX .BAS       BOX2 .BAS      PRINT .BAS      STRING .BAS
IFTHEN .BAS    CASE .BAS      ASC .BAS        ULCASE .BAS
LTRIM .BAS     ABS .BAS       RANDOM .BAS
897824 Bytes free
Enter original filename (Enter to end)?

Thank you for participating !

Press any key to continue

```

٣ - اضغط على أى مفتاح للعودة إلى البرنامج. اختر New من قائمة File واكتب N لإخلاء الشاشة.

٤ - انتقل إلى الدرس التاسع والستين للاستمرار فى تسلسل التعلم.

الدرس الحادى والتسعون

دالة OCT\$

الوصف

تعيد دالة OCT\$ المكافئ الثمانى لتعبير عشري، وتكونها هو كما يلى :

OCT\$(numeric expression)

يقرب التعبير العدى لأقرب قيمة صحيحة قبل التحويل، فإذا كانت القيمة المقربة خارج المدى الصحيح (من 0 إلى 32,767) فتنحول هذه القيمة إلى رقم صحيح طويل قبل تقويمها، والقيمة التى تعود تكون من نوع السلسلة ولا يمكن أن تستخدم فى الحسابات.

التطبيقات

تستخدم دالة OCT\$ فى الحصول على التمثيل الثمانى لتعبير عددي، والمبرمجون المهتمون بمثل هذا التمثيل للبيانات يجنون هذه العملية مفيدة لهم، وفيما يلى بعض الأمثلة :

```
O$ = OCT$(256*4)
PRINT "Decimal 10 is octal " OCT$(10)
```

المخرجات : الرقم العشري 10 له مكافئ ثمانى 12.

عملية تقليدية

عينة البرنامج فى هذه العملية توضح دالة OCT\$. لاختبار ذلك ابدأ بتحميل بيسك السريع
١ - اكتب البرنامج التالى :


```

File Edit View Search Run Debug Calls
<Untitled>
This is a demonstration of OCT$ statement
CLS
FOR A% = 33 TO 128
  PRINT CHR$(A%); " "; OCT$(ASC(CHR$(A%))); "/ ";
NEXT

```

Immediate

Main: <Untitled> Context: Program not running 00000000

٢ - نفذ البرنامج ولاحظ استخدام OCT\$ في البرنامج.

```

! 41/ " 42/ * 43/ $ 44/ % 45/ & 46/ ' 47/ ( 50/ ) 51/ M 52/ + 53/ , 54/ - 55/ .
56/ / 57/ 0 60/ 1 61/ 2 62/ 3 63/ 4 64/ 5 65/ 6 66/ 7 67/ 8 70/ 9 71/ : 72/ ; 73
/ < 74/ = 75/ > 76/ ? 77/ @ 100/ A 101/ B 102/ C 103/ D 104/ E 105/ F 106/ G 107
/ H 110/ I 111/ J 112/ K 113/ L 114/ M 115/ N 116/ O 117/ P 120/ Q 121/ R 122/ S
123/ T 124/ U 125/ V 126/ W 127/ X 130/ Y 131/ Z 132/ [ 133/ \ 134/ ] 135/ ^
136/ _ 137/ ` 140/ a 141/ b 142/ c 143/ d 144/ e 145/ f 146/ g 147/ h 150/ i 151
/ j 152/ k 153/ l 154/ m 155/ n 156/ o 157/ p 160/ q 161/ r 162/ s 163/ t 164/ u
165/ v 166/ w 167/ x 170/ y 171/ z 172/ { 173/ | 174/ } 175/ ~ 176/ ` 177/ g
280/

```

Press any key to continue

٣ - اضغط على أى مفتاح للعودة إلى البرنامج. من قائمة File اختر New واكتب N لإخلاء الشاشة.

٤ - انتقل إلى الدرس السابع للاستمرار فى تسلسل التعلم.

الدرس الثاني والتسعون

عبارة ON event GOSUB

الوصف

تستخدم عبارة ON event GOSUB في اصطلياد خطأ حدث والتأثير على تنقيح البرنامج طبقاً لنتائج مثل هذه الأحداث. وتكوين عبارة ON event GOSUB هو كما يلي :

ON event GOSUB line number : line label

يعنى الخطأ الرأسى ان يكون رقم السطر line number أو اسم السطر line label للبرنامج الفرعى محدداً. جزء event هو الوحدة التى توجه لحدث معين. وفيما يلى الأنواع المختلفة للأحداث.

الحدث	الوصف
TIMER (n)	يختبر وحدة TIMER ويميز متى تمر n ثانية. يقع الرقم n بين 1 و 86,400.
COM (n)	يختبر وحدة COM بالنسبة للرموز التى يتم تلقيها عند بوابة الاتصالات n. والرقم n إما أن يكون 1 أو 2.
PLAY (n)	يختبر صف الموسيقى PLAY لعدد الرموز المتروك للعب. ويحدث الحدث عندما يصبح طول الصف أقل من n.
KEY (n)	يختبر ما إذا كان المفتاح المذكور على أنه n فى عبارة KEY مضغوطاً أم لا.
PEN	يختبر ما إذا كان القلم الضوئى نشطاً أم لا.
STRING (n)	يختبر عصا الحركة وما إذا كانت مضغوطاً عليها أم لا. الرقم n هو القادح الذى يكون مضغوطاً.

يتم عمل الاصطلياد وابطال عمله باستخدام ما يلى :

event ON	'enables event trapping
event OFF	'disables event trapping
event STOP	'suspends event trapping

البرنامج الفرعى الذى يستدعى بواسطة عبارة GOSUB ON event ينفذ حدث STOP لمنع اصطيايد الاعادة الذاتية وينفذ حدث ON عند انتهاء البرنامج الفرعى. أى حدث يحدث بينما يكون الحدث STOP نشطاً يتم تذكرته وتشغيله عندما تنفذ عبارة الحدث ON.

التطبيقات

تستخدم عبارة GOSUB ON event فى تطبيقات البرمجة المطورة. واصطيايد وتشغيل مثل هذه الأحداث يكون مفيداً جداً عندما يكون من اللازم عمل تحكم شديد على بيئة التنفيذ. فإيجاد ما إذا كانت المفاتيح مضغوطة عليها أم لا وماذا يحدث عند احدى بوابات الاتصالات وماذا يحدث عند الوحدات الأخرى وتوجيه عبارات خلفية الموسيقى والمقدرة على استخدام هذه المعلومات لزيادة الدقة فى منطق البرنامج تمثل الاستخدام الأساسى لعبارات GOSUB ON event. وعلى بعد ذلك أمثلة لعبارات GOSUB ON event.

يكشف أول مثال متى يقل عدد الملاحظات المتروكة فى خلفية صف الموسيقى عن 10 ثم يقفز عند ذلك إلى البرنامج الفرعى PlayContinue.

```
PLAY ON
ON PLAY(10) GOSUB PlayContinue
..
PLAY ..
PlayContinue:
..
RETURN
```

ويمكن المثال التالى من اصطيايد مفتاح الوظيفة F3، اصطيايد متى يكون المفتاح F3 مضغوطة عليه، ثم يقفز بعد ذلك إلى البرنامج الفرعى AbortJob.

```
KEY (3) ON
ON KEY(3) GOSUB AbortJob
..
AbortJob:
INPUT "Are you sure ? (Y/N) ";Yn$
..
RETURN
```

عملية تقليدية

هذه العملية عبارة عن مثال لكيفية استخدام عبارة GOSUB ON event. ابدأ بتحميل
بيسك السريع.

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls F1=Help
<Untitled>
' This program demonstrates the use of the ON KEY(n) GOSUB statement.
' Turn the trapping on for function key F1. Trap the event and
' branch to GetOut. Branch from GetOut to Finit, and end the program.

KEY(1) ON
ON KEY(1) GOSUB GetOut
CLS

DO
  InCh$ = INKEY$
LOOP UNTIL True

Finit:
END

GetOut:
PRINT "You did it! You finally found out how to stop this program."
BEEP: BEEP
RETURN Finit

----- Immediate -----
Main: <Untitled> Context: Program not running 000.00-000

```

٢ - نفذ البرنامج. ولايقافه اضغط على F1. لاحظ كيفية استخدام عبارة GOSUB ON event
في البرنامج لتعريف مفتاح محدد وتشغيل هذا الحدث. اضغط على أى مفتاح للعودة إلى
البرنامج.

٣ - من قائمة File اختر Save. اكتب ONEVENT. BAS كاسم للملف وحدد أن شكل الملف
نصى واحفظ هذا البرنامج.

٤ - انتقل إلى الدرس الثالث والتسعين للاستمرار في تسلسل التعلم.

الدرس الثالث والتسعون

عبارات ON.. GOTO و ON.. GOSUB

الوصف

تتسبب عبارات ON.. GOTO و ON.. GOSUB فى التفرع إلى رقم سطر محدد أو اسم سطر محدد طبقاً للقيمة المعطاة فى التعبير، وتكوينها هو كما يلى :

```
ON expression GOTO line1,line2
ON expression GOSUB line1,line2
```

جزء expression هو أى تعبير عددي ينتج عنه رقم صحيح كنتيجة له، وعندما لا تكون النتيجة رقماً صحيحاً فتقرب إلى أقرب رقم صحيح قبل تنفيذ GOTO أو GOSUB، ويتفرع البرنامج إلى line1 إذا ما كانت نتيجة التعبير 1 وإلى line2 إذا ما كانت نتيجة التعبير 2 وهكذا، وينفذ البرنامج السطر التالى عندما تكون نتيجة التعبير 0 أو أى رقم أكبر من عدد أرقام أو أسماء الأسطر الموجودة فى العبارة، وتنتج رسالة خطأ بحدوث استدعاء غير سليم لدالة عندما ينتج عن التعبير رقماً سالباً أو رقماً أكبر من 25.

الجزئين ON.. GOTO و ON.. GOSUB هما أرقام أو أسماء أسطر يتفرع إليها البرنامج طبقاً لقيمة التعبير العددي، وهناك حد ضمنى لعدد أرقام أو أسماء الأسطر الذى يتبع عبارة ON.. GOTO و ON.. GOSUB وهو 255، ويمكن أن تخط أرقام الأسطر وأسماء الأسطر فى قائمة واحدة.

والفرق بين ON.. GOTO و ON.. GOSUB يقع فى نوع التفرع الذى ينفذ، ففي ON.. GOTO يكون التفرع شبيهاً بعبارة GOTO بدون امكانية ذاتية للعودة إلى العبارة التى تودى إلى التفرع، أما فى ON..GOSUB فيكون التفرع مثل عبارة GOSUB ويتوقع وجود عبارة RETURN للعودة إلى العبارة التى أدت إلى التفرع.

التطبيقات

تقدم عبارات ON.. GOTO و ON.. GOSUB طريقة قوية أخرى لعمل قرارات التفرع فى البرنامج، وهى تمثل عبارة IF.. THEN.. ELSE متعددة المستويات أو عبارة CASE STATE

MENT وذلك فى صورة أكثر ايجازاً فقط، وتستخدم عبارات ON.. GOTO و ON.. GOSUB فى المواقف التى تكون النتائج المختبرة فيها بسيطة ولا تتطلب تكوين تفرع متعدد المستويات أكثر تطوراً، وفيما يلى بعض الأمثلة :

مثال ١

```
ON Se% GOTO 120,130,150,200,1000
```

مثال ٢

```
ON Fahrenheit GOTO Frozen,Freezing,VeryCold,Cold,Thawing
```

مثال ٣

```
ON Result GOSUB SimpleInterest,CompoundInterest,NothingForYou
```

مثال ٤

```
Input "Enter selection ":Choice%
ON Choice% GOTO FileSelect,EditRecord,NewRecord,1200
```

عملية تقليدية

هذه العملية توضح استخدام عبارة ON..GOSUB، ابدأ بتحميل ببسك السريع.

١ - اكتب البرنامج التالى :

```
File Edit View Search Run Debug (Alt+F5) F1-Help
Untitled
CLS
PRINT : PRINT "Centigrade TO Fahrenheit TO Centigrade ..."
PRINT : INPUT "1 -> C to F / 2 -> F to C": Fc
ON Fc GOSUB CZF, FZC
END

CZF:
INPUT "Enter temperature in Centigrade ": Cent
PRINT : PRINT "Temperature in Centigrade ": Cent:
PRINT "in Fahrenheit ": (9 * Cent + 160) / 5
RETURN

FZC:
INPUT "Enter temperature in Fahrenheit ": Farhn
PRINT : PRINT "Temperature in Fahrenheit ": Farhn:
PRINT "in Centigrade ": (5 * Farhn - 160) / 9
RETURN

: Immediate
```

٢ - نفذ البرنامج. لاحظ استخدام عبارة GOSUB ON.. في البرنامج. اكتب 2 واضغط على مفتاح الادخال. اكتب 32 واضغط على مفتاح الادخال. اضغط على أى مفتاح للعودة إلى البرنامج.

```
Centigrade TO Fahrenheit TO Centigrade ....
1 -> C to F / 2 -> F to C 2
Enter temperature in Fahrenheit ? 32
Temperature in Fahrenheit 32 in Centigrade 0
```

Press any key to continue

٣ - من قائمة File اختر New واكتب N لإخلاء الشاشة.

٤ - انتقل إلى الدرس الواحد والأربعين للاستمرار في تسلسل التعلم.

الدرس الرابع والتسعون

عبارة OPEN

الوصف

تعد عبارة OPEN للمدخلات أو للمخرجات في أو من ملف أو وحدة، ويجب أن يكون الملف أو الوحدة مفتوحاً قبل أى محاولة ادخال مدخلات أو الحصول على مخرجات باستخدامه. وتحدد عبارة OPEN ذاكرة احتياطية للمدخلات أو المخرجات من الملف أو الوحدة وتحدد حالة الاتصال المستخدم مع الذاكرة الاحتياطية. وهناك صيغتان لعبارة OPEN. وتكوين كل منهما هو كما يلي:

التكوين الأول :

OPEN file FOR mode ACCESS access lock AS # filename LEN=recien

جزء file هو تعبير سلسلة يحدد الوحدة اختياريًا واسم الملف أو اسم المسار طبقاً لاصطلاحات DOS لتسمية الملفات والمسارات. جزء FOR mode يرمز إلى نوع النشاط المراد تنفيذه على الوحدة أو الملف. والصيغ المختلفة للحالة هي كما يلي :

الحالة	المعنى
OUTPUT	مخرجات تتابعية.
INPUT	مدخلات تتابعية.
APPEND	مخرجات تتابعية. تضع مؤشر الملف عند نهاية الملف وتعد لاضافة مخرجات من هذه النقطة. ويوضح رقم السجل لآخر سجل.
RANDOM	اتصال عشوائي. وهذه هي الحالة التقليدية. عندما لا يوجد جزء اتصال فيحاول ببسك السريع أن يفتح الملف أو الوحدة ثلاث مرات في الترتيب التالي :
	- قراءة وكتابة.
	- قراءة فقط.
BINARY	اتصال بملف ثنائي. يسمح بالاتصال بالملف أو بالوحدة على مستوى البايت. يستخدم عبارات GET و PUT. وعندما لا يوجد جزء ACCESS فيحاول ببسك السريع فتح الملف ثلاث مرات مثل حالة RANDOM.

جزء access فى ACCESS يعطى نوع العملية المسموح بها على الملف أو الوحدة. ويعمل ذلك فقط على وحدات تعمل تحت صيغ DOS التى تدعم استخدام الشبكات (الصيغة 3.0 وما بعدها). برنامج بدء الشبكة (SHARED. EXE) ينفذ ليسمح لى تسهيلات اغلاق وعندما تستخدم مع صيغ DOS قديمة تظهر رسالة بأن السمات المطورة غير متاحة. وعندما يتحدد الاتصال للملف أو لوحدة مفتوحاً بالفعل ولا تتفق طريقة الاتصال الجديدة معه فتظهر رسالة بعدم امكانية الفتح. وطرق الاتصال المختلفة هى ما يلى :

الاتصال	المعنى
READ	قراءة فقط.
WRITE	كتابة فقط.
READ WRITE	قراءة وكتابة. وهذا صحيح مع حالات RANDOM و BINARY و APPEND فقط.

يستخدم جزء lock فى بيئة التشغيل المتعدد. وهذا يتحكم فى الاتصال بالملف أو الوحدة داخل النظام. وفيما يلى الموصفات المختلفة لجزء الاغلاق :

الاغلاق	المعنى
Default	عندما لا يتحدد اغلاق فيكون الملف أو الوحدة متاحاً لهذه العملية فقط. وبقيّة البرامج التى تنفذ فى نفس الوقت يكون لها اتصال مرفوض.
SHARED LOCK READ	أى برنامج فى النظام يمكنه الاتصال بهذا الملف أو هذه الوحدة. لا يمكن قراءة الملف أو الوحدة بواسطة أى عملية أخرى. ويعمل ذلك إذا لم تكن هناك مواصفة اغلاق تعطى اتصال READ للملف أو الوحدة فقط.
LOCK WRITE LOCK READ WRITE	لا يمكن الكتابة فى الملف أو الوحدة بواسطة أى عملية أخرى. وتطبق نفس القيود عليها مثل LOCK READ. لا يمكن القراءة أو الكتابة من وإلى الملف أو الوحدة بواسطة أى عملية أخرى. وتطبق نفس القيود عليها مثل LOCK READ و LOCK WRITE.

جزء AS# يعطى رقماً (رقم ملف filenum) كرقم للملف والذي يقع بين 1 و 255. ويصاحب هذا الرقم الملف طالما أن الملف مفتوح ويتم الاتصال بالملف باستخدام هذا الرقم. جزء LEN = يعطى حجم السجل (طول السجل reclen) بعدد الرموز. بالنسبة للملفات الاتصال التتابعى يكون حجم السجل التقليدى 512 وبالنسبة للملفات الاتصال المباشر يكون حجم السجل التقليدى 128. وهناك حد على طول السجل وهو 32,767 بايت. عندما تكون حالة الملف ثنائية BINARY فيهمل هذا الجزء. يمكن أن تختلف أحجام السجلات داخل الملف التتابعى ولا يحتاج جزء طول السجل أن يتوافق مع أى حجم سجل فردى آخر.

التكوين الثانى :

OPEN mode, # filenum, file, reclen

وجزء mode فى هذا التكوين هو أحد الأجزاء التالية :

المعنى	الحالة
مخرجات تتابعية.	O
مدخلات تتابعية.	I
اتصال عشوائى.	R
ملف ثنائى.	B
اتصال تتابعى. تضع مؤشر الملف عند انتهاء الملف ورقم سجل لآخر سجل.	A

وتتأخر المكونات الأخرى العناصر الموجودة فى التكوين الأول. وهذه الصيغة لعبارة OPEN لا توفر تسهيلات اتصال ومشاركة كما هو الحال فى المكون الأول وهى معدة لأغراض توافقية عمل الصيانة مع الصيغ القديمة للبيسك.

والوحدات المدعومة كجزء من جزء الملف file هى كما يلى :

KYBRD: لوحة المفاتيح، مدخلات فقط
SCRN: موجه أو شاشة، مخرجات فقط.

COMn: بوابة الاتصالات، n هو رقم البوابة، مدخلات ومخرجات.
LPTn: طابع أسطر، n هو رقم الطابع، مخرجات فقط.
CONS: شاشة، مدخلات ومخرجات.

كما يسمح كذلك بالوحدات التي يعرفها المستفيد. وتفصيل أكثر عن مثل هذه الوحدات أفحص دليل DOS.

التطبيقات

عبارة OPEN تكون في الأساس متركزة مع تشغيل الملف في البرنامج. أى برنامج يتطلب ملفات على قرص خارجى أو ينقل معلومات إلى وحدات يجب أن يستخدم عبارة OPEN. وتستخدم الملفات في كل التطبيقات تقريباً. وفيما يلي بعض الأمثلة :

التكوين الأول :

```
OPEN "Client.Dat" FOR RANDOM AS #3
```

يفتح هذا المثال ملفاً له الاسم "Client.Dat" للاتصال العشوائى مع الملف رقم 3.

```
OPEN "Welcome.Txt" FOR OUTPUT AS #9
OPEN "Scratch.Pad" FOR INPUT AS #11
OPEN "Temp.Buf" FOR APPEND AS #1
OPEN "Network.Dir" FOR RANDOM ACCESS READ LOCK WRITE AS #100
```

التكوين الثانى :

```
OPEN "O" , 1, "Address"
```

يفتح هذا المثال ملفاً له الاسم "Address" لمخرجات مع الملف رقم 1.

```
OPEN "R", 10, "Client.Ndx"
OPEN "B", 11, "Register.Exe"
```

عملية تقليدية

هذه العملية توضح استخدام عبارة OPEN. ابدأ بتحميل ببسك السريع.

١ - أكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls F1=Help
<Untitled>
This program demonstrates the OPEN statement.

CLS
ON ERROR GOTO FileError
PRINT : PRINT "Demonstration of the OPEN statement ..."
FILES "777.*"
PRINT : INPUT "Enter filename to view: "; FileName$

IF FileName$ <> "" THEN
    OPEN FileName$ FOR INPUT AS #1
    PRINT : PRINT "Listing of file "; FileName$

    DO WHILE NOT EOF(1)
        LINE INPUT #1, InLine$
        PRINT InLine$
    LOOP

END IF

END

FileError:
PRINT "Abnormal (?) program termination.", ERR, ERL

```

Immediate

Main: <Untitled> Context: Program not running 00036:004

٢ - نفذ البرنامج. لاحظ استخدام عبارة OPEN في البرنامج. تبين الشاشة دليلاً مختلفاً به ملفات مختلفة. اكتب اسم الملف الذي حفظته اثناء تسلسل التعلم. لاحظ أن الملف يجب أن يكون في شكل ASCII لكي يعمل البرنامج بطريقة صحيحة. لقد استخدمنا الملف ASC.BAS في هذا المثال.

```

Demonstration of the OPEN statement ...
C:\QB
    <DIR> .. <DIR> BC .EXE QB .EXE
LIB .EXE QB .HLP QB .LIB QB .QLB
QB .PIF QB .BI BOX .BAS QB .OBJ
BOX .EXE EXE .MAP ASC .BAS ABS .BAS
UP .TXT 002 003 004
005 U .DAT SUB .BAS

2869248 Bytes free

Enter filename to view: ? ASC.BAS

Listing of file ASC.BAS
Num$ = "1234": Num% = 0
FOR i = 1 TO LEN(Num$)
    Num% = Num% + ((ASC(MID$(Num$, i, 1)) - 48) * (10 ^ (LEN(Num$) - i)))
NEXT i
PRINT Num$, Num%

Press any key to continue

```

- ٣ - اضغط على أى مفتاح للعودة إلى البرنامج، اختر Save من قائمة File واكتب OPEN. BAS كاسم للملف وحدد أن شكل الملف نصي واحفظ هذا البرنامج.
- ٤ - اختر New من قائمة File ثم اكتب N لإخلاء الشاشة.
- ٥ - انتقل إلى الدرس الخامس والأربعين للاستمرار في تسلسل التعلم.

الدرس الخامس والتسعون

عبارتا COM و OPEN COM

الوصف

عبارة OPEN COM : تفتح عبارة OPEN COM قناة اتصالات مع وضع قيمة ابتدائية لها بالنسبة للمدخلات والمخرجات، وتكوين عبارة OPEN COM هو كما يلي :

OPEN "COMn: oplist1 oplist2" FOR mode AS #filenum LEN=Reclen

جزء COMn هو اسم قناة الاتصالات المستخدمة مثل : COM1 أو : COM2.

جزء oplist1 له الشكل التالي : سرعة، تعادل، بيانات، توقف، ويجب أن تتبع المحتويات هذا الترتيب المحدد وعندما تكون المحتويات محنوفة فيجب أن تستخدم الفواصل في تحديد الأماكن، ويصف الجدول التالي جزء oplist1.

الخيار	الوصف
Speed	عدد البت في الثانية كمعدل لنقل البيانات (معدل بود).
Parity	N = لاشيء و E = زوجي و 0 = فردي و S = سرعة و M = علامة.
Data	عدد بت البيانات، 5 أو 6 أو 7 أو 8.
Stop	عدد بت التوقف، 1 أو 1.5 أو 2.

وفيما يلي قائمة بمحتويات جزء m. oplist2 في القائمة لها قيمة تقليدية 1000.

الخيار	الوصف
ASC	يفتح الوحدة في حالة ASCII. تتسع Tab وتجبر على عودة العربية عند انتهاء كل سطر، Ctrl-Z بدلاً من EOF ويستخدم اتفاق XON/XOFF.
BIN	يفتح الوحدة في الحالة الثنائية وهي الحالة التقليدية.
CDm	تضع سطر Data Carrier Detect في التعليق بعد m ميللي ثانية.
CSm	تضع سطر Clear To Send في التعليق بعد m ميللي ثانية.
LF	تسمح بطباعة ملف اتصالات على الطابع. ويجبر رمز تغذية السطر على الظهور بعد عودة العربية.
OPm	تحدد طول العبارة التي تنتظر لعملية فتح ناجحة.
RBn	تحدد حجم الذاكرة الاحتياطية المستقبلية بأنها n بايت. القيمة التقليدية هي 512 بايت.
RS	تضغط اكتشاف Request To Send.
TBn	تحدد حجم الذاكرة الاحتياطية للنقل بأنها n بايت. القيمة التقليدية هي 512 بايت.

و جزء mode هو أحد ما يلي والقيمة التقليدية هي RANDOM :

الحالة	الوصف
OUTPUT	مخرجات تتابعية.
INPUT	مدخلات تتابعية.
RANDOM	حالة اتصال عشوائي.

جزء filenum # هو رقم الملف المستخدم في فتح الوحدة. و جزء LEN يعطى طول السجل. عندما تفتح الوحدة على أنها RANDOM فإن LEN يساوى الذاكرة الاحتياطية للاتصال العشوائي. والقيمة التقليدية لجزء LEN هي 128 بايت.

وعندما تنفذ عبارة OPEN COM فإنها تؤدي الأشياء التالية :

- تحدد ذاكرات احتياطية وتمكن من الازعاجات interrupts.
- تحدد أن DTR مرتفع.
- عندما يكون خيار OP أو DS غير صفري فنتنظر حتى يصبح DTR مرتفعاً أو حتى يكون هناك تعليق فإذا كان هناك تعليق فتفشل OPEN COM.
- تحدد أن RTS مرتفع إذا كان خيار RS محنواً.
- عندما يكون خيار OP أو CD غير صفري فنتنظر حتى يصبح DTR مرتفعاً أو حتى يكون هناك تعليق فإذا كان هناك تعليق فتفشل OPEN COM.
- عندما تفشل عبارة OPEN COM فيلغى تحديد الذاكرات الاحتياطية وتلغى مقدرة الازعاجات interrupts وتحمى اسطر التحكم.
- عبارة COM : تمكن عبارة COM أو لا تمكن أو توقف من اصطلياد الأحداث على الوحدة COMn. وتكوينها هو كما يلي :

COM(n) ON
COM(n) OFF
COM(n) STOP

جزء n في الثلاث عبارات كلها هو رقم وحدة اتصالات مثل 1 أو 2. وتمكن عبارة COM(n) ON من اصطلياد الأحداث لبوابة اتصالات محددة. والأحداث التي تحدث عند بوابة الاتصالات تكتشف وتمرر إلى المقطع الخاص بمعالجتها. وتلغى COM(n) OFF من مقدرة اصطلياد الأحداث لبوابة محددة. الأحداث التي تحدث عند البوابة يتم تذكيرتها وتشغيلها بعد تنفيذ تسلسل COM(n) ON.

ويجب تنشيط اصطلياد الأحداث بعبارة GOSUB COM (n) ON قبل أن يمكن استخدام أى عبارة من عبارات COM (n).

التطبيقات

تستخدم عبارات OPEN COM و OFF, ON COM (n) و STOP في فتح قناة اتصالات ونقل واستقبال بيانات على وحدة COM. وفيما يلي أمثلة لعبارات COM (n) و COM.


```

OPEN "COM1: 300,E,7,1,ASC" AS #2
COM(1) ON
ON COM(1) GOSUB GetCharCom
..
OPEN "COM1: 1200,N,8,1,BIN" AS #1
..
GetCharCom:
    GET #1, ..
    ..
    RETURN

```

عملية تقليدية

حيث إن المعلومات عن إجراءات الاتصالات لا يمكن افتراضها وأن الموضوع يقع خارج مدى هذا الكتاب فلن يحتوى هذا القسم على مثال.

انتقل إلى الدرس السابع والستين للاستمرار في تسلسل التعلم.

الدرس السادس والتسعون

عبارة OPTION BASE

الوصف

تعرف عبارة OPTION BASE الحد السفلى التقليدي لمنظومة. وتكوينها هو كما يلي :

OPTION BASE n

جزء n اما أن يكون 1 أو 0. وهذا يعرف الحد السفلى لكل المنظومات الموجودة في البرنامج وتستخدم عبارة OPTION BASE مرة واحدة فقط في أحد الأجزاء وتظهر على مستوى كتابة الجزء فقط. وتستخدم عبارة OPTION BASE قبل تحديد أبعاد أى منظومة. وعندما تتصل البرامج ببعضها البعض فيكتسب البرنامج الذى يتم التوصيل إليه اعداد عبارة OPTION BASE للبرنامج الذى يتم توصيله به.

التطبيقات

عبارة OPTION BASE غير ضرورية فى بيسك السريع. وعبارة DIM لها مقدرة على تحديد حدود سفلى للمنظومات اثناء تحديد الأبعاد وليس عليها قيود عبارة OPTION BASE (هى 0 و 1). ومع ذلك ففيما يلي بعض الأمثلة بغرض التوضيح.

مثال ١

OPTION BASE 1
DIM Q(12)

مثال ٢

OPTION BASE 0
DIM WorkLst(10,10)

عملية تقليدية

هذه العملية توضح استخدام عبارة OPTION BASE. ابدأ بتحميل بيسك السريع.

١ - اختر Open وحمل البرنامج REDIM.BAS.

٢ - عدل البرنامج ليتفق مع القائمة التالية :

```

File Edit View Search Run Debug Collis F1=Help
<Untitled>
' The following program is the program from Modula 115, modified to
' demonstrate the OPTION BASE statement.

OPTION BASE 1
Max = 15
DIM A(Max)
GOTO Start

LoadArray:
  FOR Cnt = 1 TO Max
    READ A(Cnt)
  NEXT
  RETURN

FindMinMax:
  MinVal = A(1); MaxVal = A(1)
  FOR Cnt = 2 TO Max
    IF MinVal > A(Cnt) THEN
      MinVal = A(Cnt)
    END IF
    IF MaxVal < A(Cnt) THEN
      MaxVal = A(Cnt)
    END IF
  NEXT
  RETURN

Start:
  GOSUB LoadArray
  GOSUB FindMinMax
  PRINT "First pass"
  PRINT "Minimum of array: "; MinVal, "Maximum of array: "; MaxVal
  READ Max
  REDIM A(Max)
  GOSUB LoadArray
  GOSUB FindMinMax
  PRINT "Second pass"
  PRINT "Minimum of array: "; MinVal, "Maximum of array: "; MaxVal

DATA 12,23,33,43,1,56,98,656,323,44,9,88,67,54,18
DATA 18
DATA 8,89,76,54,23,32,12,4,33,54

Immediate

```

٣ - نفذ البرنامج. لاحظ استخدام عبارة OPTION BASE في البرنامج.

```
First pass
Minimum of array: 1      Maximum of array: 656
Second pass
Minimum of array: 4      Maximum of array: 89
```

Press any key to continue

- ٤ - ارجع إلى البرنامج واختر New دون أن تحفظ البرنامج.
- ٥ - انتقل إلى الدرس الحادى والسبعين للاستمرار فى تسلسل التعلم.

الدرس السابع والتسعون

عبارة PAINT

الوصف

تملأ عبارة PAINT مناطق رسومات بلون معين أو بنمط معين. وتكوينها كما يلي :

`PAINT STEP (x,y).paint.border.background`

جزء STEP الاختيارى يستخدم فى تحديد أن الاحداثيات المعطاة نسبية إلى موضع الشاشة الحالى. جزء (x,y) هو احداثى نقطة الرسم والذى يقع داخل المنطقة المراد تلوينها. ويجب أن تقع الاحداثيات المعطاة داخل أو خارج المنطقة المراد تلوينها وليس على حدودها. يمكن أن يكون جزء paint من النوع العددي أو السلسلة. وعندما يكون عددياً فيجب أن تكون قيمة خاصية اللون من النوع الصحيح. أما إذا كان سلسلة فلا تكسو عبارة PAINT والتي تملأ المنطقة بنمط معين بدلاً من لون واحد. جزء border هو اللون المستخدم فى تعريف حدود الشكل وعندما يرسم لون لحدود فيتوقف التلوين. وجزء background هو قيمة سلسلة تحدد الخلفية التي يجب تركها أثناء دهان الشكل.

الخلفية : تلون الخلفية بنمط معين بدلاً من لون واحد. وطريقة وصف هذا النمط هي كما يلي:

- ١ - ارسم النمط على شبكة من 8 أعمدة ويمكن أن يصل عدد صفوفها إلى 64.
- ٢ - ترجم كل صف إلى سلسلة من 0 و 1 واكتب ذلك.
- ٣ - حول السلسلة من 0 و 1 إلى قيم سادسة عشرية.
- ٤ - أنتج سلسلة بدمج نتيجة دالة CHR\$ للقيم السادسة عشرية التي تم الوصول إليها فى الخطوة السابقة.
- ٥ - لون الفراغ الموجود داخل الشكل مستخدماً خيار tile من عبارة PAINT.

التطبيقات

تستخدم عبارة PAINT فى تلوين أشكال الرسومات. وهذا مفيد بالنسبة للعديد من التطبيقات مثل الألعاب والتوضيحات والتقديمات. وفيما يلي مثال لعبارة PAINT.

```
SCREEN 1
COLOR 1,2
CLS
DRAW ..
CIRCLE (100,120),30
PAINT (100,120),3
```

عملية تقليدية

هذه العملية توضح استخدام عبارة PAINT. استمر إذا ما كانت لديك امكانيات رسومات ملونة فقط. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :

```

FILE  EDIT  Open  Search  Run  Debug  Call's  F1-Help
PAINT.BAS
'this program illustrates the use of the PAINT statement
'set screen mode
SCREEN 1
CLS

'define a pattern
pattern1$ = CHR$(120) + CHR$(0) + CHR$(120) + CHR$(0) + CHR$(120) + CHR$(0)
'define another pattern
pattern2$ = CHR$(&H80) + CHR$(&H60) + CHR$(&H30) + CHR$(&H6) + CHR$(&H1)

'draw a box
LINE (40, 40)-(300, 150), , B

'paint the box with the defined pattern
PAINT (50, 50), pattern2$

Immediate

```

٢ - نفذ البرنامج. لاحظ استخدام عبارة PAINT في البرنامج.

٣ - ارجع إلى البرنامج مع اخلاء الشاشة بون أن تحفظ هذا البرنامج.

٤ - انتقل إلى الدرس المائة وثمانية للاستمرار في تسلسل التعلم.

الدرس الثامن والتسعون

عبارتا PALETTE USING و PALETTE

الوصف

تغير عبارة PALETTE الألوان في مجموعة الألوان. ومجموعة الألوان عبارة عن مجموعة الألوان المستخدمة حالياً في حالة شاشة معينة. وتغير عبارة PALETTE الألوان مستبدلة إياها بمجموعة ألوان جديدة. وتكوينها هو كما يلي :

```
PALETTE attr,color
PALETTE USING arrayname(index)
```

جزء attr هو الخاصية المراد تغييرها. وجزء color هو اللون الذي يحل محل اللون الحالي لهذه الخاصية وهو تعبير عددي صحيح. ويقدم جزء USING ميزة إضافية في أنه يستخدم منظومة أرقام صحيحة أو منظومة أرقام صحيحة طويلة في استبدال مدى كامل للخواص -attrib- utes. جزء arrayname هو اسم المنظومة وجزء index هو دليل المنظومة التي يبدأ من عندها الاستبدال.

وتعمل عبارة PALETTE مع النظم التي بها بطاقات EGA أو VGA أو MCGA فقط.

التطبيقات

تستخدم عبارة PALETTE في رسم مجموعة من قيم الألوان بخاصية مستخدمة حالياً في حالة شاشة معينة. وتستخدم حالة الشاشة مجموعة من الألوان في العرض وبالرغم مما إذا كان يسمح النظام بالألوان مختلفة فعبارة PALETTE يمكن أن تستخدم في اختيار قيم ألوان للخواص معطية بذلك تحكماً أكثر في ألوان العرض. عندما تستخدم عبارة PALETTE تتغير كل الألوان المعروضة حالياً على الفور إلى مجموعة الألوان الجديدة وتستخدم المخرجات التالية مجموعة الألوان الجديدة. ويساعد جزء USING في اعداد مدى كامل للخواص مع عبارة PALETTE واحدة. وكل المحتويات السالبة باستثناء 1- تكون قيم ألوان غير صحيحة في المنظومة المستخدمة في هذا الغرض. ويقوم 1- في المنظومة بترك الخاصية دون تغيير. استخدام عبارة COLOR بعد عبارة PALETTE يضع قيماً تقليدية لألوان الخواص. وفيما يلي بعض أمثلة لعبارة PALETTE.

مثال ١

```
SCREEN 8
PALETTE 1,4
..
```

مثال ٢

```
PALETTE 4,2
..
```

مثال ٣

```
DIM P(15)
FOR C = 1 TO 15
    P(C) = C
NEXT C
..
PALETTE USING P(0)
```

يستخدم آخر مثال منظومة أعداد صحيحة في وضع مجموعة الألوان الجديدة لقيم الخاصة. لاحظ أن المنظومة لها 15 عنصراً، وهذا هو أقل حجم لمنظومة تستخدم في هذا الغرض نظراً لأنه مسموح بحد أدنى 16 لوناً في حالة بطاقة EGA وبطاقة VGA. ومع امكانية بطاقة VGA وبطاقة MCGA تستخدم منظومة اعداد صحيحة طويلة لأن قيم الألوان أكبر لهذه الحالات.

في حالة VGA نحسب الألوان بطريقة مختلفة فاللون عبارة عن قيمة لدرجات مختلفة من الألوان الأزرق والأحمر والأخضر في الشاشة وفي المطبع. والصيغة هي ما يلي :

$$\text{Color} = 65536 * \text{blue} + 256 * \text{green} + \text{red}$$

حيث blue و green و red هي قيم لشدة اللون تقع بين 0 و 63.

عملية تقليدية

حيث إنه لا يمكن افتراض أنك لديك امكانية بطاقة EGA أو بطاقة VGA أو بطاقة MCGA فلا يحتوى هذا القسم على مثال. ويجب أن يوجد المزيد من المعلومات عن امكانية الألوان في نظامك في دليل المستفيد الخاص بالجهاز نفسه.

انتقل إلى الدرس المائة والثامن والخمسين للاستمرار في تسلسل التعلم.

الدرس التاسع والتسعون

عبارة PCOPY

الوصف

تنسخ عبارة PCOPY صفحة من الشاشة في مكان آخر، وتكوينها هو كما يلي :

PCOPY n1,n2

جزء n1 هو الصفحة المراد نقلها وجزء n2 هو الصفحة المراد النقل إليها. ولناقشة عدد الصفحات المرئية المتاحة في حالات مختلفة للشاشة ارجع إلى الدرس الرابع والعشرين.

التطبيقات

يمكن استخدام عبارة PCOPY في عديد من التطبيقات والتي تكون مخرجات الشاشة فيها أقصى ما يمكن وتكون سرعة المخرجات حاسمة. يمكن عرض إحدى صفحات العرض أثناء اعداد صفحة أخرى. وتطبيقات الرسومات هي اختيار حتمي كما يمكن لتطبيقات النصوص الموجهة للحالة text-mode-oriented أن تتمتع من زيادة سرعة مخرجات الشاشة. وفيما يلي أمثلة لعبارة PCOPY:

```

FILE  EDIT  Open  Search  Run  Debug  Calls  F1-Help
<Untitled>
' This program demonstrates the use of the PCOPY statement.

COLOR 2, 1
CLS
'Write a bunch of asterisks
FOR i = 1 TO 100
  PRINT "x";
NEXT

'Copy the current page to page 1
PCOPY 0, 1

CLS
'Write a bunch of dashes
FOR i = 1 TO 1000
  PRINT "-";
NEXT

'Copy this to page 2
PCOPY 0, 2
  
```

```

Copy back first from page 1, pause, then from page 2
FOR i = 1 TO 3
  PCOPY 1, 0
  FOR t = 1 TO 500: NEXT
  PCOPY 2, 0
  FOR t = 1 TO 500: NEXT
NEXT

```

Immediate

Main: (Untitled) Context: Program not running 000.36:005

عملية تقليدية

- هذه العملية تستخدم البرنامج المقدم في قسم التطبيقات من هذا الدرس. استمر إذا كانت لديك إمكانيات عمل رسومات ملونة لدعم ذلك. ابدأ بتحميل ببسك السريع.
- ١ - اكتب البرنامج الموجود في قسم التطبيقات في هذا الدرس.
 - ٢ - نفذ البرنامج. لاحظ المخرجات واستخدام عبارات PCOPY في البرنامج.
 - ٣ - احفظ البرنامج كملف نصي تحت اسم BAS. PCOPY مع اخلاء الشاشة.
 - ٤ - انتقل إلى الدرس المائة وواحد للاستمرار في تسلسل التعلم.

الدرس المائة

دالة PEEK وعبارة POKE

الوصف

تعمل دالة PEEK وعبارة POKE مع ذاكرة الكمبيوتر وتكمل كل منهما الأخرى.
دالة PEEK : تعيد دالة PEEK البايت عند موقع محدد للذاكرة، وتكوينها هو كما يلي :

PEEK(address)

جزء address هو قيمة تقع في المدى من 0 إلى 65,535. تعيد دالة PEEK قيمة عديدة صحيحة تقع في المدى من 0 إلى 255. والقطاع الذي يكون فيه الفرع هو القطاع المحدد بواسطة عبارة DEF SEG. وفي غياب عبارة DEF SEG يستخدم قطاع بيانات بيسك السريع.
عبارة POKE : تكتب عبارة POKE بيانات في موقع محدد من الذاكرة، وتكوينها كما يلي :

POKE address, byte

جزء address هو تعبير عددي يقع في المدى من 0 إلى 65,535 وجزء byte هو تعبير عددي يقع في المدى من 0 إلى 255. والعنوان يكون الفرع بنفس الطريقة مثل دالة PEEK.

التطبيقات

تستخدم PEEK في قراءة معلومات معينة من ذاكرة الكمبيوتر وفي اتخاذ قرارات مبنية على ذلك. وتستخدم POKE في الكتابة في مواقع الذاكرة مباشرة، واحد التطبيقات هو الرسومات والرسومات المتحركة مثل المباريات. وفيما يلي مثال لذلك.

```
'This program demonstrates how the POKE function is used for simple animation.
'On non IBM PC's, or full compatibles this program may not work.
'Set the segment to the monochrome screen starting address.
'Use &HB800 for color monitors
DEF SEG = &HB800
CLS
FOR x = 1 TO 10
'POKE the asterisk; the multiplication with 160 causes the char. to move downwards
POKE x * 160, 42
wait
FOR l = 1 TO 300: NEXT
'POKE a space where the asterisk was
POKE x * 160, 32
NEXT x
'Reset the default segment
DEF SEG
```

المخرجات من هذا المثال تبين نجمة تتحرك من قمة الشاشة لأسفل.

عملية تقليدية

هذه العملية توضح استخدام PEEK و POKE. ابدأ بتحميل ببسك السريع،

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls F1=Help
<Untitled>
' This is a demonstration of the PEEK statement. The program writes the
' alphabet at the top of the screen, finds the letter Q and changes it
' to another character.
' On non-IBM PC's, or full compatibles this program may not work.
' Set the default segment to the monochrome screen starting address.
' Use &H8000 for color monitors.
DEF SEG = &H8000
CLS
PRINT "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

FOR x = 1 TO 2048

' PEEK at what is there; convert that to a character.
IF CHR$(PEEK(x)) = "Q" THEN
' POKE the ASCII character 42 (an asterisk) where Q was, beep, and leave.
POKE x, 42
BEEP
EXIT FOR
END IF
NEXT

' Reset the default segment
DEF SEG

Immediate

Main: <Untitled> Context: Program not running 00000001

```

٢ - نفذ البرنامج. لاحظ استخدام PEEK و POKE في البرنامج.

```

ABCDEFGHIJKLMNOPQRSTUVWXYZ

Press any key to continue

```

٢ - ارجع إلى البرنامج. احفظ البرنامج كملف نصي تحت اسم POKE.BAS مع اخلاء الشاشة.

٤ - انتقل إلى الدرس التاسع للاستمرار في تسلسل التعلم.

الدرس المائة وواحد

دالة PEN

الوصف

تعطى دالة PEN إحداثيات القلم الضوئى، وتكوينها هو كما يلى :

$$PEN(n)$$

حيث n قيمة تقع فى المدى من 0 إلى 9، ويعطى الجدول التالى القيم التى تعود لكل قيمة من قيم n.

N	القيمة التى تعود
0	1- إذا تحرك القلم لأسفل من عند آخر استخدام لدالة PEN وإلا فإن القيمة تكون 0.
1	احداثى x لنقطة الرسم لأخر استخدام للقلم.
2	احداثى y لنقطة الرسم لأخر استخدام للقلم.
3	1- إذا كانت حركة القلم الحالية لأسفل و 0 إذا كانت لأعلى.
4	آخر احداثى x لنقطة الرسم يكون صحيحاً.
5	آخر احداثى y لنقطة الرسم يكون صحيحاً.
6	وضع صف الشاشة لأخر قلم مستخدم.
7	وضع عمود الشاشة لأخر قلم مستخدم.
8	وضع صف الرمز لأخر قلم مستخدم.
9	وضع عمود الرمز لأخر قلم مستخدم.

التطبيقات

تستخدم دالة PEN فى قراءة احداثيات القلم الضوئى وهى محددة لهذه الوحدة فقط. ولا تعمل دالة PEN مع مشغل الفأرة لأن مشغل الفأرة يستخدم نفس استدعاءات BIOS مثل دالة PEN، وفيما يلى بعض الأمثلة :

```
PEN ON
PPos = PEN(3)
IF PPos < 0 THEN PRINT "Pen down" ELSE PRINT "Pen up"

PEN ON
FOR PP = 0 TO 9
    PRINT PEN(PP)
NEXT PP
```

يطبع آخر مثال كل القيم الممكنة التي تعود بواسطة دالة PEN. لاحظ كذلك عبارات PEN ON في المثال وهذا ضروري لأن القلم الضوئي يبدأ في التوقف ويجب أن يعاد قبل أن يمكن استخدام دالة PEN. وتظهر رسالة خطأ بأن هناك استدعاء دالة خطأ إذا لم يحدث ذلك.

عملية تقليدية

حيث إنه لا يمكن افتراض استخدام وإتاحة القلم الضوئي فلا يقدم هذا القسم مثلاً. انتقل إلى الدرس المائة واثنين للاستمرار في تسلسل التعلم.

الدرس المائة واثنان

عبارات PEN ON و PEN OFF و PEN STOP

الوصف

تقوم عبارات PEN ON و PEN OFF و PEN STOP بتمكين وعدم تمكين والغاء تصيد حدث القلم الضوئى، وأساسيات استخدام كل عبارات الاحداث ON و OFF و STOP هى نفس الأساسيات، وتتعامل هذه المجموعة من العبارات مع القلم الضوئى، للمزيد من المعلومات عن اصطيات الأحداث أرجع إلى الدرس الثانى والتسعين. وتكوين عبارات PEN ON و PEN OFF و PEN STOP هو كما يلى :

```
PEN ON
PEN OFF
PEN STOP
```

تمكن عبارة PEN ON من اصطيات الاحداث للقلم الضوئى. وتستخدم العبارة (n) PEN و GOSUB/GOTO فى التأثير على أحد الاحداث عند اصطياتها.

وتلغى PEN OFF مقدرة اصطيات الاحداث للقلم الضوئى، والاحداث التى تحدث بعد تنفيذ هذه العبارة لا يمكن تذكرها ولا يمكن تشغيلها بعبارات PEN ON تأتى بعد ذلك، وتوقف عبارة PEN STOP اصطيات الاحداث للقلم الضوئى، والاحداث التى تحدث بعد تنفيذ هذه العبارة يمكن تذكرها ويمكن تشغيلها بعد تنفيذ تسلسل من عبارات PEN ON.

التطبيقات

تطبيقات PEN ON و PEN OFF و PEN STOP محدودة ومحددة لاستخدام القلم الضوئى، ويفلق القلم الضوئى وينشط اصطيات الاحداث بعبارة (n) ON PEN. يجب أن تنفذ عبارة PEN ON قبل أن يمكن استخدام دالة PEN.

انتقل إلى الدرس المائة والثامن والثلاثين للاستمرار فى تسلسل التعلم.

الدرس المائة وثلاثة

دالة وعبارة PLAY

الوصف

الكلمة المحجوزة PLAY يمكن أن تستخدم بطريقتين، كدالة وعبارة. كدالة فإنها تعيد عدد النوت الموسيقية الموجودة في الصف إذا كانت هناك موسيقى في الخلفية وإلا فتكون نتيجتها صفراً. وعبارة فإنها تلعب موسيقى كما هو محدد بسلسلة المؤشر. وتكوينها في كل من الحالتين هو:

تكوين الدالة :

PLAY(n)

هذه هي دالة PLAY. ويكون n هو مؤشر صوري يمكن أن يكون أي قيمة موسيقية. تعيد PLAY عدد النوت الموسيقية المتروكة في صف موسيقى الخلفية في المتغير n.

تكوين العبارة :

PLAY string expression

هذه هي عبارة PLAY. يحتوي جزء string - expression على أوامر للموسيقى. ويصف الجدول التالي الأوامر المسموح بها :

الأمر	الآثر
>	يزيد هذا من الثمانية octave بمقدار 1. وأعلى ثمانية هي 6.
<	يقلل هذا من الثمانية بمقدار 1. وأقل ثمانية هي 0.
o number	يحدد هذا الثمانية والمدى من 0 إلى 6. ويرقم الثمانية من C إلى B أي ثمانية واحدة تصاعديا encompasses CDEDEFAGB والقيمة التقليدية للثمانية هي 1.
N number	يلعب النوتة الموسيقية المعطاة بالرقم. الرقم يقع بين 0 و 84 (وتكون كل النوت الموسيقية في سبعة ثمانيات). عندما يكون الرقم 0 فهذا يعني راحة.

الأمر	الأثر
A - G	يلعب نوتة موسيقية بين A و G كما هو محدد، ويمكن أن يتبع النوتة # أو + لتعني حاداً sharp أو - لتعني مسطحاً flat.
L number	يحدد مدى استمرار النوت الموسيقية، ويقع الرقم في المدى من 1 إلى 64 حيث 1 يمثل نوتة كاملة و 4 تمثل ربع نوتة وهكذا، عندما تتطلب نوتة معينة طولاً مختلفاً فليتبّع النوتة طولها، مثال ذلك G12 تكافئ L12G.
MN	تحدد أن الموسيقى معتادة، كل نوتة تلعب 7/8 من الطول المحدد بواسطة L.
ML	تحدد أن الموسيقى متسقة، كل نوتة تلعب الطول المحدد بواسطة L.
MS	تحدد أن الموسيقى متقطعة، كل نوتة تلعب 3/4 الطول المحدد بواسطة L.
P number	تحدد توقفاً لحظياً في الموسيقى، يقع الرقم بين 1 و 64 مناظراً لطول النوتة المحدد بواسطة L.
T number	تحدد السرعة للموسيقى، ويقع العدد بين 32 و 255 وله قيمة تقليدية 120، والسرعة هي عدد ربع النوت التي تلعب في دقيقة واحدة.
MF	تحدد أن الموسيقى تلعب في الأمامية، ويظهر صوت كل نوتة بعد انتهاء آخر نوتة، وهذا هو الوضع التقليدي.
MB	تحدد أن الموسيقى تلعب في الخلفية، ويستمر تنفيذ البرنامج اثناء لعب النوتة، ويمكن لعب 32 نوتة كحد أقصى في الخلفية في نفس الوقت، ويتسبب ذلك في لعب النوتة 3/2 من الطول المحدد بواسطة L مضروباً في أعداد السرعة بواسطة T، يمكن استخدام فترات متعددة بعد النوتة، وتضيف كل فترة طولاً مساوياً 1/2 طول الفترة السابقة للنوتة، (وهذا يكافئ «النوتة المنقطة» باصطلاحات الموسيقى).
"X"+VARP-TRE\$ (string variable)	تنفذ السلسلة الموجودة في متغير السلسلة.

التطبيقات

يمكن استخدام دالة PLAY فى التأكد من تقدم الاحداث فى موسيقى الخلفية واتخاذ قرارات بتنفيذ البرنامج. وتستخدم عبارة PLAY فى لعب الموسيقى. وفيما يلى بعض الأمثلة :

دالة PLAY : فيما يلى أمثلة لدالة PLAY :

```
NotesLeft% = PLAY(N%)
ON PLAY(10) GOSUB FillerUp
```

عبارة PLAY : فيما يلى أمثلة لعبارة PLAY

```
PLAY "o0ABolCDEFG P2 GFEDCo0BA"
PLAY "o2 AA..BB..o3CC > AA..BB..o4CC"
PLAY "T180 o2 P2 P8 L8 GGG L2 E-"
St$ = "T180 o2 p2 p8 l8 ggg l2 e-"
PLAY "X" + VARPTR$(St$)
```

عملية تقليدية

العملية التالية توضح عبارة PLAY. ابدأ بتحميل ببسك السريع.

١ - اكتب البرنامج التالى :

```
File Edit View Search Run Debug Calls FI-Help
<Untitled>
This program demonstrates the PLAY statement
CLS
KEY OFF
PLAY "MBL16T155"
FOR I = 1 TO 4
  READ MS
  PRINT MS
  PLAY MS
NEXT I
KEY ON
END
DATA "o4e8e8e4e8e8e4e8g8c8d8eZ
DATA "f8f8f8f8f8e8e8eZ6eZ6e8d8d8e8d4g4
DATA "e8e8e4e8e8e4e8g8c8d8eZ
DATA "f8f8f8f8f8e8e8eZ6eZ6e8g8f8d8cZ

Immediate

Main: <Untitled> Context: Program not running MMI: 0.0
```

٢ - نفذ البرنامج، لاحظ الموسيقى من عبارة PLAY. لاحظ استخدام عبارة PLAY فى البرنامج. اضغط على أى مفتاح للعودة إلى البرنامج.

```
o4e8e8e4e8e8e4e8g8c8d8eZ
f8f8f8f8f8e8e8e26e26e8d8d8e8d4g4
e8e8e4e8e8e4e8g8c8d8eZ
f8f8f8f8f8e8e8e26e26g8g8f8d8c2
```

Press any key to continue

٣ - من قائمة File اختر Save. اكتب PLAY.BAS كاسم للملف واحفظ هذا البرنامج كملف نصي.

٤ - انتقل إلى الدرس المائة وأربعة للاستمرار فى تسلسل التعلم.

الدرس المائة وأربعة

عبارات PLAY ON و PLAY OFF و PLAY STOP

الوصف

تستخدم عبارات PLAY ON و PLAY OFF و PLAY STOP في التحكم في اصطيات الأحداث في البرنامج. وتمكن عبارة PLAY ON من اصطيات الأحداث وتلغى عبارة PLAY OFF إمكانية اصطيات الأحداث وتوقف عبارة PLAY STOP من اصطيات الأحداث. وتستخدم هذه العبارات بالاتصال مع عبارة ON PLAY، وتكوينها هو كما يلي :

PLAY ON
PLAY OFF
PLAY STOP

تستخدم عبارة PLAY ON في ايجاد المقدرة على اصطيات الأحداث بحيث إنه يميز الحدث المراد اختباره عندما يحدث. ومثال للحدث هو عدد النوت الموسيقية المتروك في موسيقى الخلفية، وتؤخذ القرارات عند حدوث الحدث.

وتستخدم عبارة PLAY OFF في الغاء المقدرة على اصطيات الأحداث ولا يمكن تذكرة الأحداث التي تحدث.

وتستخدم عبارة PLAY STOP في ايقاف اصطيات الأحداث ولا ينفذ اجراء معالج الأحداث. تنفذ PLAY STOP تلقائياً داخل اجراء معالج الأحداث بحيث أن الأحداث الأكثر لاتتسبب في الاعادة الذاتية للبرنامج. والأحداث التي تحدث بعد عبارة PLAY STOP يمكن تذكرها وتشغيلها عندما تنفذ عبارة PLAY ON.

يحدث اصطيات الأحداث لعبارة PLAY عندما تلعب الموسيقى في الخلفية فقط.

التطبيقات

تستخدم عبارات PLAY ON و PLAY OFF و PLAY STOP عندما يكون مطلوباً تحكماً شديداً على حدوث الأحداث أثناء تنفيذ البرنامج. وتستخدم عبارات PLAY ON و PLAY OFF و PLAY STOP في اصطيات الأحداث أثناء تنفيذ عبارة PLAY. وفيما يلي مثال لذلك.

```

PLAY ON
PLAY "GE GE > FD "
..
IF PLAY(10) THEN GOSUB TenLeft
..
ON PLAY(3) GOSUB PlaySomeMore
..
PlaySomeMore:
    PLAY NewStr$
    ..
    PLAY OFF
RETURN
    
```

عملية تقليدية

هذه العملية توضح استخدام عبارات PLAY ON و PLAY OFF و PLAY STOP. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls F1=Help
<Untitled>
' This program demonstrates the PLAY ON and PLAY OFF statements.
CLS
PLAY ON
PLAY "MBL16T128"
READ N$
PLAY N$

Cont:
ON PLAY(3) GOSUB NextNote

NextNote:
    READ N$
    PLAY N$
    PLAY OFF
    END

DATA "f8f8f8f8f8e8e8e26e26e8d8d8e8d4g4
DATA "f8f8f8f8f8e8e8e26e26e8g8g8f8d8c2
    
```

Immediate

Main: <Untitled> Context: Program not running 00010:037

٢ - نفذ البرنامج. لاحظ استخدام عبارات PLAY ON و PLAY OFF و PLAY STOP في البرنامج. اضغط على أى مفتاح للعودة إلى البرنامج.

٣ - من قائمة File اختر N لإخلاء الشاشة.

٤ - انتقل إلى الدرس الثانى والتسعين للاستمرار فى تسلسل التعلم.

الدرس المائة وخمسة

دالة PMAP

الوصف

ترسم دالة PMAP أحداثيات معينة لأحداثيات منطقية أو واقعية مبنية على مفتاح الدالة. وتكونها هو كما يلي :

PMAP(expression,function)

جزء expression هو الأحداثيات المراد رسمها وجزء function هو المفتاح الذي يعد للرسم. ويبين الجدول التالي مفاتيح الدالة المسموح بها وما تفعله.

مفتاح لدالة	مفعله
0	يرسم أحداثيات منطقية لإحداثى X واقعى.
1	يرسم أحداثيات منطقية لإحداثى Y واقعى.
2	يرسم أحداثيات واقعية لإحداثى X منطقى.
3	يرسم أحداثيات واقعية لإحداثى Y منطقى.

التطبيقات

دالة PMAP مفيدة جدا عندما تستخدم مع عبارة WINDOW لأن دالة PMAP يمكنها أن ترسم الأحداثيات الموجودة بنظام الأحداثيات السابقة التعريف. وفيما يلي مثال لدالة PMAP :

```
SCREEN 1
WINDOW SCREEN (10,50)-(150,100)
x = 12: y = 1
x = PMAP(x,2)
y = PMAP(y,3)
```

```
SCREEN 1
COLOR 1,2
CLS
DRAW ..
DRAW ..
Xp = POINT[0]: Yp = POINT(1)
DRAW ..
DRAW "BM =" +VARPTR$(Xp) + " ,=" +VARPTR$(Yp)
DRAW ..
```

يوضح المثال كيفية استخدام دالة POINT في حفظ الاحداثيات عند نقطة معينة أثناء انتاج الرسومات. وتستخدم عبارة DRAW الاحداثيات المحفوظة للعودة إلى هناك والاستمرار في الرسم.

عملية تقليدية

المثال الموجود في الدرس الخامس والثلاثين، DRAW، هو توضيح جيد لدالة POINT. وقد حفظ البرنامج تحت اسم DRAW.BAS. راجع هذا البرنامج لرؤية احدى طرق استخدام دالة POINT.

انتقل إلى الدرس السابع والتسعين للاستمرار في تسلسل التعلم.

الدرس المائة وستة

دالة POINT

الوصف

تعيد دالة POINT إما رقم اللون من نقطة الرسم أو إحداثيات نقطة الرسم. وتكوينها هو كما يلي :

POINT (x,y)
POINT (number)

يعطى التكوين الأول قيمة اللون عند إحداثيات نقطة الرسم (x,y) وعندما تكون الإحداثيات خارج المدى فتعيد POINT القيمة -1. ويعيد التكوين الثانى إحداثيات نقطة الرسم الحالية طبقاً لقيمة الرقم. ويعطى الجدول التالى القيم والإحداثيات التى تعيدها الدالة :

الرقم	الإحداثيات التى تعيدها الدالة
0	إحداثى x الواقعى.
1	إحداثى y الواقعى.
2	إحداثى x المنطقى.
3	إحداثى y المنطقى.

الإحداثى المنطقى هو الإحداثى النسبى لعبارة WINDOW النشطة حالياً.

التطبيقات

دالة POINT مفيدة فى الحصول على إحداثيات نقطة الرسم الحالية وذلك أثناء رسم الصور. والوظائف المختلفة التى تسمح بها دالة POINT فى الحصول على إحداثيات نقطة الرسم تحدث بأكثر من طريقة واحدة. وفيما يلى مثال لدالة POINT :

```
SCREEN 1
COLOR 1,2
CLS
DRAW ..
DRAW ..
Xp = POINT[0]: Yp = POINT(1)
DRAW ..
DRAW "BM =" +VARPTR$(Xp)+",=" +VARPTR$(Yp)
DRAW ..
```

يوضح المثال كيفية استخدام دالة POINT في حفظ الاحداثيات عند نقطة معينة أثناء انتاج الرسومات. وتستخدم عبارة DRAW الاحداثيات المحفوظة للعودة إلى هناك والاستمرار في الرسم.

عملية تقليدية

المثال الموجود في الدرس الخامس والثلاثين، DRAW، هو توضيح جيد لدالة POINT. وقد حفظ البرنامج تحت اسم DRAW.BAS. راجع هذا البرنامج لرؤية إحدى طرق استخدام دالة POINT.

انتقل إلى الدرس السابع والتسعين للاستمرار في تسلسل التعلم.

الدرس المائة وسبعة

دالة POS

الوصف

تعطى دالة POS الوضع الحالى لنقطة البداية. وتكوينها هو كما يلى :

```
POS(column)
```

يستخدم جزء column إلا أنه يهمل. وتعيد هذه الدالة الموقع الأفقى الحالى لنقطة البداية.

التطبيقات

تستخدم دالة POS فى الحصول على معلومات عن عمود نقطة البداية وتستخدم هذه الدالة فى توفير تحكم أفضل للشاشة فى الحالة التالية ولتعزيز السطح البينى للمستفيد ببرامج تطبيقات. وفيما يلى بعض أمثلة لدالة POS.

```
IF POS(0) > 20 THEN
LOCATE CSRLIN + 1, 1
END IF

CROW = POS(0)
LOCATE 12, CROW+1
```

عملية تقليدية

البرنامج المستخدم فى هذه العملية هو نفس البرنامج المستخدم فى الدرس الخامس والعشرين. ويوضح البرنامج بحرية استخدام دالة POS فى تطبيق عملى. ابدأ بتحميل بيسك السريع.

١ - حمل البرنامج المستخدم فى الدرس الخامس والعشرين والمسمى CSRLIN.BAS.

٢ - أنظر إلى الدالة GetChar بالضبط على Shift-F2. لاحظ استخدام دالة POS في حفظ موقع الصف الحالي.

٣ - ارجع إلى البرنامج مع اخلاء الشاشة نون أن تحفظ هذا البرنامج.

٤ - انتقل إلى الدرس الستين للاستمرار في تسلسل التعلم.

الدرس المائة وثمانية

عبارة PRESET

الوصف

ترسم عبارة PRESET نقطة على الشاشة عند الاحداثيات المحددة. وتكوينها هو كما يلي:

```
PRESET STEP x,y,color
```

يحدد جزء STEP أن الاحداثيات نسبية إلى الوضع الحالى وهو جزء اختياري. ويعطى جزء (x,y) احداثيات الشاشة للنقطة المراد رسمها. ويصف جزء color خاصية اللون للنقطة واستخدام هذا الجزء اختياري. وعندما يحذف فيكون اللون المختار هو اللون التقليدي للخلفية. أى يصبح السطر غير مرئياً.

التطبيقات

عبارة PRESET مفيدة فى رسم نقاط على الشاشة تكون مستقلة عن أشياء الرسومات الموجودة بالفعل على الشاشة. مثال ذلك أنه يمكن استخدامها فى رسم نجوم فى السماء أو فى اضافة مقطع إلى الرسم. كما يمكن استخدامها كذلك فى رسم رسومات الخطوط. والاستخدامات محدودة بالتطبيق الذى يعد له البرمجة. وفيما يلي مثال لعبارة PRESET.

```
FOR C = 1 TO 100
  PRESET(C,100), 2
NEXT
```

عملية تقليدية

هذه العملية توضح استخدام عبارة PRESET. استمر إذا كانت لديك امكانيات رسومات ملونة تدعم ذلك فقط. ابدأ بتحميل بيسك السريع.

١ - أكتب البرنامج التالى :

```

File Edit View Search Run Debug Calls F1=Help
PRESET.BAS
' This program uses the PRESET statement.
SCREEN 1: COLOR 1, 3
CLS

FOR cnt = 1 TO 50
    PRESET (65, cnt), 2

NEXT
' Change the viewport and draw another line.
VIEW (100, 10)-(150, 30), , 1

FOR cnt = 1 TO 50
    PRESET (1, cnt), 2

NEXT

Immediate

Name: PRESET.BAS Context: PRESET.BAS 00002:002

```

٢ - نفذ البرنامج. لاحظ استخدام عبارة PRESET في البرنامج.

٣ - ارجع إلى البرنامج واحفظ البرنامج كملف نصي تحت اسم PRESET.BAS.

٤ - انتقل إلى الدرس المائة واثنى عشر للاستمرار في تسلسل التعلم.

الدرس المائة وتسعة

عبارة PRINT

الوصف

تستخدم عبارة PRINT فى عرض بيانات عديدة أو بيانات سلاسل على الشاشة. وتكونها هو كما يلى :

PRINT expression list|.|;

جزء expression list هو تسلسل من عناصر البيانات المراد طباعتها وتكون مفصولة عن بعضها البعض بواسطة فواصل أو فواصل منقوطة. وتكون البيانات العددية مشكلة طبقاً لنوعها المحدد وطباعتها. ودائماً ما توضع بيانات السلسلة بين علامتى تنصيص مزدوجتين فى عبارة PRINT. وشكل عبارة PRINT لطباعة كل من نوعى البيانات موضح أدناه.

النوع	تعبير لطباعة	المخرجات
عددى صحيح	PRINT 20 PRINT - 20 PRINT 20 * 2	20 -20 40
دقة فردية	PRINT 2.1E-6 PRINT 2.1E-7	0000021 2.1E-7
(تطبع اعداد الدقة الفردية حتى 7 خانات للأرقام) دقة مزدوجة	PRINT 2.1D-15 PRINT 2.1-16	.00000000000000021 2.1D-16
(تطبع أعداد الدقة المزدوجة حتى 16 خانة للأرقام) سلسلة	PRINT "Dead on Arrival"	Dead on Arrival

ويتحقق تشكيل المخرجات عن طريق استخدام الفراغات أو الفواصل المنقوطة.
وتأثير كل تشكيل مذكور فى الجدول التالى :

المخرجات	عبارة PRINT
1 2	PRINT 1, 2
(تسبب الفاصلة فى تجزئة السطر إلى منطقتى طباعة كل منها يشغل ١٤ خانة وتطبع كل قيمة فى قائمة التعبير عند بداية المنطقة التالية)	
1 2	PRINT 1 2
1 2	PRINT 1;2
1 2	PRINT 1; : PRINT 2
(المخرجان من عبارة PRINT يوضعان على نفس السطر بسبب استخدام الفاصلة المنقوطة)	
1	PRINT 1 : PRINT 2
2	
(توضع قيمة مخرجات عبارة PRINT الثانية فى السطر التالى)	

رموز التشكيل التى سبق وصفها تحدد موقع ظهور عناصر البيانات على الشاشة.

التطبيقات

الاستخدام المناسب لعبارة PRINT مع رموز التشكيل المناسبة يعد وسيلة قوية. وفى برنامج العينة الموجود فى الدرس الثالث يستخدم أمر PRINT فى رسم مستطيل وإخراج الرسالة داخل هذا المستطيل. وفيما يلى أمثلة أخرى لاستخدام عبارة PRINT.

```
PRINT "New baby !", "WOW ! !"
```

وتشبه المخرجات ما يلى :

```
New baby !      WOW ! !
PRINT SQR(2)
```


المخرجات :

```
1.414214
FOR Cnt = 1 TO 40: PRINT ' ' : NEXT Cnt
```

المخرجات :

.....

عملية تقليدية

البرنامج التالي يوضح استخدام عبارة PRINT. يشتمل متغير السلسلة على رموز تشكيل يستخدمها البرنامج في زيادة اضاءة نص معين. ولحاولة ذلك ابدأ بتحميل بيك السريع.

١ - اكتب البرنامج التالي :

```
File Edit View Search Run Debug Calls F1=Help
<Untitled>
OutStr$ = "Hello @USER!, This is your @friendly! computer !"
SPos = 1: CLS : COLOR 15, 0
FOR Cnt = 1 TO LEN(OutStr$)
  IF MID$(OutStr$, SPos, 1) = "@" THEN COLOR 0, 15: SPos = SPos + 1
  IF MID$(OutStr$, SPos, 1) = "!" THEN COLOR 15, 0: SPos = SPos + 1
  PRINT MID$(OutStr$, SPos, 1); : SPos = SPos + 1
NEXT
Immediate
Main: 'Untitled' Context: Program not running 000007:005
```

٢ - اضغط على Shift-F5 لتنفيذ البرنامج فترى ما يلي :

Hello **USER**. This is your **friendly** computer !

Press any key to continue

٣ - اضغط على أى مفتاح للعودة إلى قائمة البرنامج.

٤ - من قائمة File اختر Save واكتب PRINT.BAS كاسم للملف ثم اختر الشكل النصي لحفظ هذا البرنامج.

٥ - من قائمة File اختر New.

٦ - انتقل إلى الدرس الخامس عشر للاستمرار فى تسلسل التعلم.

الدرس المائة وعشرة

عبارة PRINT USING

الوصف

تستخدم عبارة PRINT USING في تشكيل مخرجات سلاسل ومخرجات عددية بطريقة محددة، وتكوينها هو كما يلي :

PRINT USING format string; expression

جزء format string هو تعبير سلسلة توجد فيه رموز التشكيل المستخدمة في طباعة البيانات، وجزء expression هو البيانات المراد طباعتها ويكون من النوع العددي أو نوع السلسلة، ويمكن طباعة أكثر من تعبير واحد باستخدام فواصل منقوطة لفصلها عن بعضها البعض، ورموز التشكيل لطباعة بيانات سلسلة تختلف عن رموز التشكيل لطباعة بيانات عددية. وفيما يلي وصف لرموز تشكيل بيانات السلسلة.

رمز التشكيل	الغرض منه
!	يتسبب في طباعة أول رمز من تعبير السلسلة فقط.
\ \	يطبع $2 + n$ رمزاً من تعبير السلسلة حيث n هو عدد الفراغات الموجودة بين الشرطتين المائتين للخلف. عندما تكون السلسلة أطول من n فتعمل الرموز الإضافية. أما إذا كانت السلسلة أقل من n فتضبط مخرجات السلسلة من ناحية اليسار مع ترك فراغات من ناحية اليمين.
&	ينسحب في طباعة تعبير السلسلة كما هو.

وفيما يلي وصف لرموز التشكيل لطباعة بيانات عددية :

رمز التشكيل	الفرض منه
#	يستخدم هذا الرمز فى تحديد موقع الرقم. فإذا كان العدد له أرقام أقل عن المواصفة فيطبع العدد مرحلاً لليمين مع وجود فراغات سابقة للعدد. أما إذا كانت أرقام العدد أكبر من الوضع المحدد فتهمل الأرقام الأكثر. يطبع علامة عشرية فى مكان ظهوره. وتقرب البيانات العددية عندما تكون هناك حاجة لذلك.
+	يسمح بطباعة اشارة العدد قبله أو بعده طبقاً لما هو محدد له.
-	يسمح بطباعة اشارة سالب.
**	يسمح بظهور نجوم فى الفراغات السابقة للعدد.
\$\$	يسمح بطباعة علامة دولار أمام القيمة العددية.
**\$	يدمج تأثير النجوم وعلامات الدولار. وتملأ الفراغات السابقة للعدد بنجوم ثم تظهر علامة الدولار أمام العدد.
,	عندما تستخدم على يسار علامة عشرية فإنها تتسبب فى طباعة فاصلة كل ثلاث خانات من على يسار العلامة العشرية. أما إذا ما استخدمت فى نهاية سلسلة التشكيل فإنها تتسبب فى طباعة فاصلة.
^^^	يحدد شكل الاس (E+xx). واستخدام خمس علامات منها يمكن من طباعة اعداد أكبر (E+xxx). ويتسبب استخدام العلامة العشرية فى ضبط الأرقام المعنوية من ناحية اليسار مع ضبط الأس.
-	هذه الشرطة التى تكون تحت حرف (فراغ فى هذه الحالة) تطبع الرمز التالى كثابت حرفى .
%	تتسبب فى طباعة اشارة النسبة المئوية عندما يكون العدد أكبر من الحقل. وعندما يقود التقريب إلى عدد يتعدى هذا الحقل فتطبع النسبة المئوية قبل العدد.

أقصى عدد للخانات التى يمكن تحديدها للأرقام هو 24. وعندما يزيد العدد عن ذلك تظهر رسالة خطأ بأن استدعاء الدالة غير صحيح.

التطبيقات

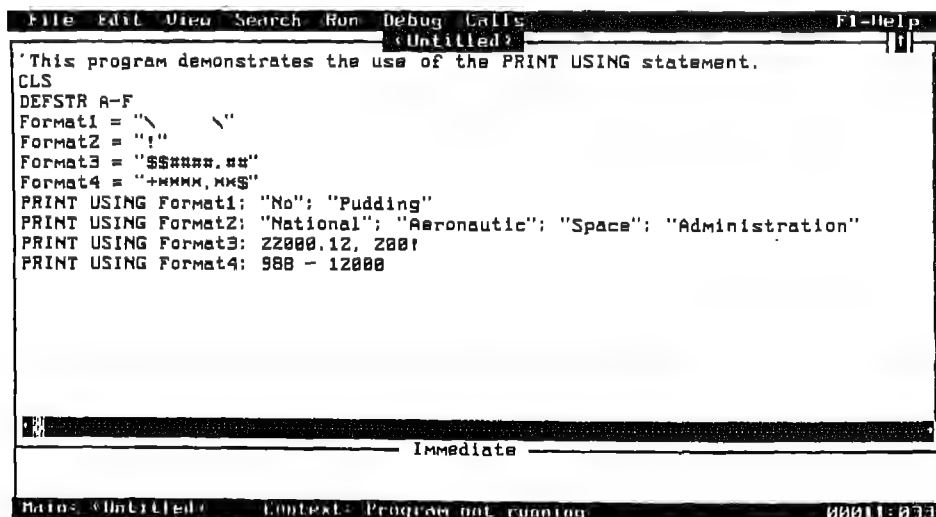
عبارة PRINT USING مريحة جداً عندما تكون هناك حاجة إلى التحكم في تشكيل بيانات المخرجات. وعبارة PRINT USING بإمكانياتها القوية للتشكيل تكون اختياراً واضحاً عند طباعة التقارير. وفيما يلي أمثلة لهذه العبارة :

```
PRINT USING "!";"Dead";"On";"Arrival"
PRINT USING "/  /";"Dead";" On"; "Arrival"
PRINT USING "&";"Dead";" On";" Arrival"
PRINT USING "####";123456
PRINT USING "*****";123
PRINT USING "$$";123.00
```

عملية تقليدية

هذه العملية توضح استخدام عبارة PRINT USING، ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :



```
File Edit View Search Run Debug Calls F1-Help
<Untitled>
This program demonstrates the use of the PRINT USING statement.
CLS
DEFSTR A-F
Format1 = "\      \"
Format2 = "!"
Format3 = "$$####.##"
Format4 = "+####.##$"
PRINT USING Format1: "No"; "Pudding"
PRINT USING Format2: "National"; "Aeronautic"; "Space"; "Administration"
PRINT USING Format3: 22000.12, 2001
PRINT USING Format4: 988 - 12000

Immediate

Main: <Untitled> Context: Program not running 0001:001
```

٢ - نفذ البرنامج، لاحظ التشكيلات المختلفة لمخرجات البرنامج، لاحظ كذلك استخدام عبارة PRINT USING في تحقيق كل من هذه التشكيلات. اضغط على أى مفتاح للعودة إلى البرنامج.

```
No      Pudding
NASA
$22000.12  $200.00
%-11012

Press any key to continue
```

٣ - من قائمة File اختر New واكتب N لإخلاء الشاشة.

٤ - انتقل إلى الدرس المائة والسادس عشر للاستمرار في تسلسل التعلم.

الدرس المائة والحادى عشر

عباراتا PRINT # USING و PRINT

الوصف

تتصرف عبارات PRINT # و PRINT # USING مثل عبارات PRINT و PRINT USING فيما عدى ان مقصد المخرجات يكون ملفا ، وتكوينها هو كما يلى :

تكوين عبارة PRINT # :

```
PRINT # filename, expression list
```

جزء filename هو رقم الملف المحدد بواسطة عبارة OPEN. جزء expression list هو قائمة بعناصر البيانات المراد كتابتها فى الملف . ارجع الى الدرس المائة وتسعة لمعرفة رموز التشكيل التى تستخدم فى هذه العبارة.

تكوين عبارة PRINT # USING :

```
PRINT # filename, USING string expression, expression list
```

جزء filename هو رقم الملف المحدد فى عبارة OPEN. وجزء USING يشبه جزء USING فى عبارة PRINT USING. وجزء string expression من جزء USING مكون بطريقة متماثلة. ويكون expression list عبارة عن قائمة عناصر البيانات المراد كتابتها بالتشكيل المقدم بواسطة جزء USING. ارجع الى الدرس المائة وعشرة لمعرفة مواصفات التشكيل المستخدمة فى جزء USING.

التطبيقات

عبارتا PRINT # و PRINT # USING ما هما الا جزء من ترسانة تشغيل الملفات القوية. واستخدامهما يقتصر على تمييز المبرمج ونوع التطبيق فقط.

عبارة PRINT # :

مثال ١

```
OPEN "Session.Log" FOR OUTPUT AS #2
PRINT #2 "Session start date "; DATE$, "time "; TIME$
```

مثال ٢

```
PRINT #3 Var1,Var2,Var3
```

مثال ٣

```
PRINT #1 S1 S2 S3
```

عبارة PRINT # USING :

مثال ١

```
OPEN "Issue.Lst" FOR RANDOM AS #10
FormatList$ = "\ \ \"
PRINT #10 USING FormatList$; "Plenipotentiary"
```

مثال ٢

```
PRINT #1 USING "!"; "Save"; "Our"; "Souls"
```

عملية تقليدية

هذه العملية توضح استخدام عبارتي PRINT # و PRINT # USING . ابدأ بتحميل
بيسك السريع .

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Help
Untitled2
This program illustrates the use of the PRINT # and PRINT # USING
statements. The program opens a file and writes data to it. The data
is provided by the program.
The file is closed and then listed to demonstrate the effect of the
PRINT # and PRINT # USING statements.

CLS
OPEN "Print.Fil" FOR OUTPUT AS #3
L1$ = "Earth Moving Equipment"; L2$ = "12"; L3$ = "$120,000"
The PRINT # statement.
PRINT #3, L1$, L2$, L3$
L1$ = "Farm Equipment"; L2$ = "22"; L3$ = "$85,000"
PRINT #3, L1$, L2$, L3$
Format$ = "\ \ \"
PRINT #3, USING Format$; L1$, L2$, L3$
PRINT #3, USING "!"; "Save", "Our", "Souls"
CLOSE #3

Immediate
Main: Untitled2 Context: Program not running 00012:000

```


- ٢ - نفذ البرنامج، لاحظ استخدام عبارات PRINT # و PRINT # USING في البرنامج.
- ٣ - لا يطبع البرنامج أى شيء على الشاشة. اختر Exit من قائمة File واكتب N للخروج من بيسك السريع. استخدم امر Type من نظام التشغيل DOS لترى محتويات الملف PRINT . FILE . وفيما يلي الملف الذى يكتب باستخدام امر Type من DOS .

```
C:\QB>type print.fil
Earth Moving Equipment      12      $120,000
Farm Equipment              22      $85,000
Farm 22      $85,0
SOS
C:\QB>
```

- ٤ - انتقل الى الدرس الثالث والستين للاستمرار فى تسلسل التعلم.

الدرس المائة والثانى عشر

عبارة PSET

الوصف

ترسم عبارة PSET نقطة على الشاشة عند احداثيات محددة. وتكوينها هو كما يلى :

```
PSET STEP (x,y),color
```

يحدد جزء STEP ان الاحداثيات نسبية للموقع الحالى وهو اختيارى، ويعطى جزء (x , y) احداثيات الشاشة للنقطة المراد رسمها . وجزء color يصف خاصية اللون للنقطة وهو اختيارى، وعندما يحذف هذا الجزء فيكون اللون المختار هو لون الامامية التقليدى.

التطبيقات

عبارة PSET مفيدة فى رسم نقاط على الشاشة بون الاعتماد على اشياء الرسومات الموجودة بالفعل على الشاشة. مثال ذلك يمكنك ان تستخدمها فى رسم نجوم فى السماء او فى اضافة مقطع للرسم . ويمكن ان تستخدم كذلك فى رسم رسومات خطوط، واستخدامها محدود بتطبيقات تكون مبرمجة فقط، وفيما يلى امثلة لعبارة PSET :

مثال ١

```
FOR C = 1 TO 100
  PSET(C,100)
NEXT
```

مثال ٢

```
FOR C = 1 TO 200
  PSET(100,INT(SIN(C)))
NEXT
```

عملية تقليدية

هذه العملية توضح استخدام عبارة PSET فى رسم رسومات . استمر اذا كانت لديك امكانيات رسومات ملونة تدعم ذلك فقط، ابدا بتحميل بيسك السريع .

١ - اختر Open وحمل البرنامج PRESET. عدل عبارات PERSET الى ما هو مبين فى القائمة التالية :

```

File Edit View Search Run Debug Calls F1=Help
PRESET.BAS
' This program uses the PSET statement.
SCREEN 1: COLOR 1, 3
CLS

FOR cnt = 1 TO 50
    PSET (65, cnt)
NEXT

VIEW (100, 10)-(150, 30), . 1
FOR cnt = 1 TO 50
    PSET (1, cnt)
NEXT

Immediate

Main: PRESET.BAS Context: Program not running C:\00012\016

```

- ٢- نفذ البرنامج . لاحظ استخدام عبارة PSET في البرنامج.
- ٣ - ارجع الى البرنامج مع اخلاء الشاشة بون ان تحفظ هذا البرنامج.
- ٤ - ارجع الى الدرس الثامن والتسعين للاستمرار في تسلسل التعلم.

الدرس المائة والثالث عشر

عبارة RANDOMIZE

الوصف

نضع عبارة RANDOMIZE القيمة الابتدائية لمنتج الارقام العشوائية وتعرف عملية تحديد القيمة الابتدائية بأنها اعادة وضع قيمة للاساس reseedng، وتكوينها هو كما يلي :

`RANDOMIZE numeric expression`

إذا حذف التعبير العددي فيتوقف البرنامج ويسأل عن قيمة بالملقن :

`Random number seed (-32768 to 32767) ?`

وإلا فيستخدم التعبير العددي في اعادة وضع قيمة للاساس لمنتج الارقام العشوائية .

التطبيقات

الفرض من استخدام عبارة RANDOMIZE هو منع دالة RAND من انتاج نفس تسلسل الارقام العشوائية . فاذا كان تسلسل الارقام العشوائية مختلفا في كل مرة ينفذ فيها البرنامج فتستخدم دالة RANDOMIZE مع مؤشر مختلف في كل تنفيذ وفيما يلي بعض الامثلة :

```
RANDOMIZE 22
RANDOMIZE 10 * LastVal%
RANDOMIZE TIMER
```

يستخدم آخر مثال القيمة التي تعود بواسطة دالة TIMER في وضع قيمة ابتدائية لمنتج الارقام العشوائية .

عملية تقليدية

العملية التالية توضح استخدام دالة RANDOMIZE، ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls F1-Help
<Untitled>
'This program demonstrates the RANDOMIZE statement.
'This program generates random numbers using the RND function.
CLS
RANDOMIZE TIMER
PRINT "Numbers generated using the RND function:"
FOR Cnt = 1 TO 20
    PRINT INT((32768 - 0 + 1) * RND(Cnt \ 1.1) + 0),
NEXT
'
Immediate
Main: <Untitled> Context: Program not running 000000:0000

```

٢ - نفذ البرنامج . لاحظ مخرجات البرنامج واستخدام دالة RANDOMIZE فى البرنامج .
 لاحظ كذلك ان الارقام العشوائية اجبرت على ان تكون ارقاما صحيحة. فى هذا البرنامج
 توضع قيمة ابتدائية لمنتج الارقام العشوائية وعلى هذا فقد تختلف مخرجاتك عن المخرجات
 التالية :

Numbers generated using the RND function:				
15646	2653	13412	6399	31523
31961	38557	8414	4399	28921
21411	6995	5111	4841	21369
29882	12236	884	21952	3851

Press any key to continue

- ٣ - اضغط على اى مفتاح للعودة الى البرنامج . من قائمة File اختر Save واحفظ هذا البرنامج كملف نص له الاسم RANDOM.BAS .
- ٤ - فى قائمة File اختر New مع اخلاء الشاشة.
- ٥ - انتقل الى الدرس المائة للاستمرار فى تسلسل التعلم.

الدرس المائة والرابع عشر

عبارة READ

الوصف

تستخدم عبارة READ في قراءة بيانات من عبارة DATA في متغيرات وتكوينها هو

READ variable list

كما يلي :

الجزء	الوصف
READ variable list	كلمة من كلمات بيسك السريع المحجوزة. قائمة متغيرات مفصولة عن بعضها البعض بفواصل وتقرأ فيها البيانات، وتتحدد القيم من عبارة DATA في هذه المتغيرات.

ويحدث خطأ تحت الشروط التالية :

• إذا كانت هناك عناصر بيانات في عبارة DATA أقل من المتغيرات الموجودة في قائمة

المتغيرات.

مثال

```
READ L1,L2,L3,L4,L5
..
DATA 100,24,234
```

• إذا لم يتفق نوع عنصر البيانات مع نوع المتغير الموجود في عبارة DATA.

مثال

```
READ Name$,AcctNo
..
DATA 92855,Cliff Brooks
```

• إذا كان عنصر بيانات عددي أكبر مما يستطيع المتغير ان يحتويه.

مثال

```
READ Month%
..
DATA 40000
```

يجب ان تقرأ عناصر متغيرات السجل عنصرا يتلو عنصر اخر فاذا كان عدد عناصر البيانات في عبارة DATA يتعدى عدد المتغيرات الموجودة في قائمة متغيرات عبارة READ فتقرأ عبارة READ التالية البيانات بدءا بأخر عنصر بيانات لم يقرأ .

مثال

```
READ Wun,Too,Thri
READ Fowr,Fif,Sicz
..
DATA 1,2,3,4,5,6
```

التطبيقات

عبارة READ مفيدة جدا في تحميل البيانات في متغيرات اثناء تنفيذ البرنامج. ودائما ما تستخدم عبارة READ مع عبارة DATA وعادة ما تستخدم مع عبارة RESTORE. وفيما يلي بعض الامثلة :

مثال ١

```
READ FName$,Minit$,LName$,Address$
DATA David,W,Sullen,"1313 Mockingbird Ln, Metropolis"
```

لاحظ ان عنصر البيانات Address\$ موضوع داخل علامتى تنصيب مزدوجة . وهذا لأن الفاصلة جزء من عنصر البيانات . يجب ان توضع البيانات من نوع السلسلة بين علامتى تنصيب مزدوجة اذا ما احتوت البيانات على فراغات سابقة او تابعة او اذا ما احتوت البيانات على نقطتين رأسييتين او فاصلة.

مثال ٢

```
READ Qt%,Pnt%,Ltr%
READ Oz%,Lb%,Ton%
DATA 12,19,99
DATA 16,1000,1
```


مثال ٢

```
DIM A(10) AS INTEGER
READ A(1),A(2),A(3),A(4),A(5),A(6),A(7),A(8),A(9),A(10)
DATA 10,9,8,7,6,5,4,3,2,1
```

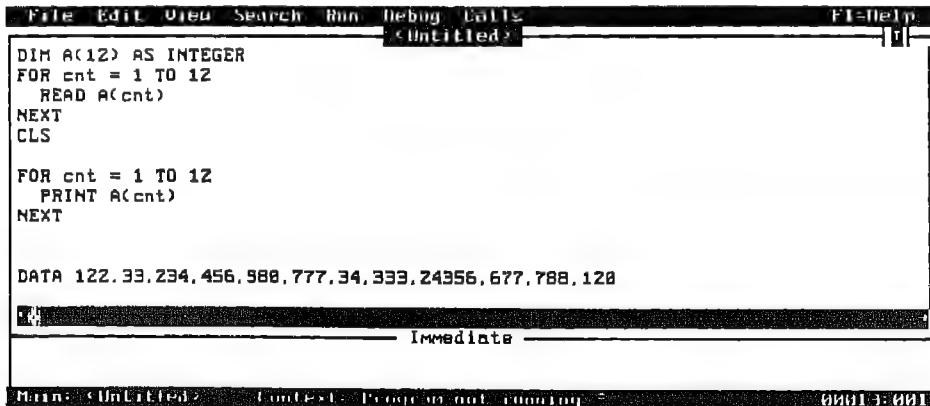
مثال ٤

```
TYPE GeneralAccount
    Name AS STRING*30
    AcctType AS STRING*3
    AcctLimit AS DOUBLE
END TYPE
DIM NewAccount AS GeneralAccount
READ NewAccount.Name,NewAccount.AcctType,NewAccount.AcctLimit
DATA ACME Gag Gifts,NEW,1000.00
```

عملية تقليدية

هذه العملية تعطى مثالا لبرنامج يستخدم عبارة READ. ابدأ بتحميل بيك السريع.

١ - اكتب البرنامج التالي :



```
File Edit View Search Run Debug Exit F1=Help
<Untitled>
DIM A(12) AS INTEGER
FOR cnt = 1 TO 12
    READ A(cnt)
NEXT
CLS

FOR cnt = 1 TO 12
    PRINT A(cnt)
NEXT

DATA 122,33,234,456,988,777,34,333,24356,677,788,128

Immediate
Main: <Untitled> Context: Program not running 0001:001
```

٢ - نفذ البرنامج ولاحظ استخدام عبارة READ في تحميل بيانات داخل متغيرات.

122
33
234
456
980
777
34
333
24356
677
788
120

Press any key to continue

- ٣ - اضغط على اى مفتاح للعودة الى البرنامج. اضغط على Alt - F ثم اضغط على مفتاح الادخال وأكتب N لإخلاء الشاشة.
- ٤ - انتقل الى الدرس السابع والعشرين للاستمرار فى تسلسل التعلم.

الدرس المائة والخامس عشر

عبارة REDIM

الوصف

تستخدم عبارة REDIM في تعديل حجم الذاكرة المحدد لاي منظومة بعبارة REDIM ويجب ان تكون المنظومة ديناميكية \$DYNAMIC وتكونها هو كما يلي :

REDIM variable (subscripts) AS type

جزء variable هو اسم متغير صحيح في بيسك السريع . وجزء (subscript) اختياري ويستخدم في تعريف ابعاد المنظومة. ويعرف جزء A Stype نوع المتغير. يمكن ان يكون النوع نوعا بسيطا (INTEGER او LONG او SINGLE او DOUBLE او STRING) او من النوع الذي يعرفه المستفيد.

جزء (subscripts) له التكوين التالي :

(lowerlimit TO upperlimit)

تتسبب عبارة REDIM في اعتبار كل المنظومات على انها ديناميكية اثناء الترجمة. اثناء التنفيذ يعاد تحديد موقع البيانات الموجودة في المنظومة قبل اعادة تحديد الابعاد وذلك بالحجم الجديد. وتفقد كل البيانات التي كانت في المنظومة قبل اعادة تحديد الحجم. وتوضع قيمة صفر للمتغيرات العددية وقيمة فراغ لمتغيرات السلسلة.

يمكن استخدام عبارة REDIM في تغيير حجم المنظومة الا انه لا يمكن استخدامها في تغيير ابعاد المنظومة. ينتج المثال التالي رسالة بأن المنظومة لها أبعاد بالفعل .

```
REM $DYNAMIC
DIM TempArr(12,12)
REIM TempArr(12,12,12)
```

وينفذ المثال التالي بدون اخطاء خاصة بابعاد المنظومة.

```
REM $DYNAMIC
DIM TempArr(12,12)
...
REDIM TempArr(10,10)
```

التطبيقات

عبارة REDIM مفيدة في التحكم في متطلبات البرنامج من الذاكرة اثناء وقت التنفيذ حيث يمكن ان تتحدد المنظومات ويلغى تحديدها كلما كان ذلك مطلوباً. عندما يكون لبرنامج متطلبات ذاكرة كبيرة وليس لديه اتصال لمثل هذه الذاكرة فيمكن ان يتمتع البرنامج بميزة الذاكرة المتاحة عن طريق تحديد والغاء تحديد المنظومات اثناء وقت التنفيذ. وفيما يلي امثلة لعبارة REDIM :

مثال ١

```
REDIM RT(34,34)
```

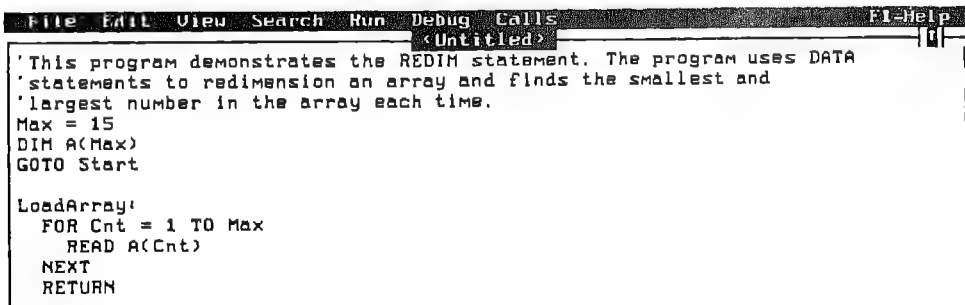
مثال ٢

```
REM $DYNAMIC
DIM A1(1000,1000)
...
REDIM A1(100,100)
```

عملية تقليدية

هذه العملية توضح استخدام عبارة REDIM. ابدأ بتحميل بيسك السريع .

١ - اكتب البرنامج التالي :



```
File Edit View Search Run Debug Calls FI-Help
<Untitled>
' This program demonstrates the REDIM statement. The program uses DATA
' statements to redimension an array and finds the smallest and
' largest number in the array each time.
Max = 15
DIM A(Max)
GOTO Start

LoadArray:
  FOR Cnt = 1 TO Max
    READ A(Cnt)
  NEXT
  RETURN
```

```

FindMinMax:
  MinVal = A(1): MaxVal = A(1)
  FOR Cnt = 2 TO Max
    IF MinVal > A(Cnt) THEN
      MinVal = A(Cnt)
    END IF
    IF MaxVal < A(Cnt) THEN
      MaxVal = A(Cnt)
    END IF
  NEXT
  RETURN

Start:
  GOSUB LoadArray
  GOSUB FindMinMax
  PRINT "First pass"
  PRINT "Minimum of array: "; MinVal, "Maximum of array: "; MaxVal
  READ Max
  REDIM A(Max)
  GOSUB LoadArray
  GOSUB FindMinMax
  PRINT "Second pass"
  PRINT "Minimum of array: "; MinVal, "Maximum of array: "; MaxVal

DATA 12,23,33,43,1,56,98,656,323,44,9,88,67,54,18
DATA 18
DATA 8,89,76,54,23,32,12,4,33,54

```

Immediate

Name: \Untitled\ Context: Program not running 000004:001

٢ - نفذ البرنامج، لاحظ استخدام عبارة REDIM في البرنامج.

```

First pass
Minimum of array: 1      Maximum of array: 656
Second pass
Minimum of array: 4      Maximum of array: 89

```

Press any key to continue

٣ - ارجع الى البرنامج واحفظه على انه ملف نص مع اعطاء الاسم REDIM.BAS له مع اخلاء الشاشة.

٤ - انتقل الى الدرس السادس والتسعين للاستمرار في تسلسل التعلم.

الدرس المائة والسادس عشر

عبارة REM

الوصف

تستخدم عبارة REM فى اضافة تعليقات الى البرنامج . ويمكن تحقيق نفس التأثير من خلال استخدام الفاصلة . وتكوينها هو كما يلى :

REM remark

جزء remark هو سطر نصى يساعد فى فهم وتتبع شفرة البرنامج . وعبارة ERM ليست عبارة للتنفيذ ولا تؤثر على تنفيذ البرنامج . ويمكن ان تظهر عبارات التنفيذ الاخرى على نفس السطر الواقعى لعبارة REM اذا ما استخدمت نقطتان رأسيتان كفاصل بينهما . كما تستخدم عبارات REM كذلك فى احتواء اشباه الاوامر (\$STATIC او \$DYNAMIC او \$INCLUDE) فى البرنامج وقد نوقشت هذه الاوامر فى الدروس الخاصة بها .

التطبيقات

عبارة REM مريحة فى تقديم ملاحظات توضيحية فى البرنامج . كما انها ضرورية كذلك فى تقديم اى واحد من اشباه الاوامر فى البرنامج . وفيما يلى بعض الامثلة :

```
REM Variable declaration
REM Stand alone process for calculating Cyclical Redundancy Check
REM Output procedure for user-defined device USR:
'Note the variable Type2 is used as an alias for TypeCast
' also note that the loop variable is different
REM $INCLUDE: "Files.Bas"
'Simple interest calculation : Int! = 0.12 : Period% = 12
```

عملية تقليدية

هذه العملية توضح استخدام عبارة REM . ابدأ بتحميل ببسك السريع .

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls F1-Help
<Untitled>
REM This program demonstrates the REM statement
REM The program illustrates the inaccuracy of computer addition.
'Notice the value of s after the 1000 iterations: it should be
'100 but is actually less than that.

CLS

FOR i = 1 TO 1000: s = s + .1: NEXT
PRINT s

Immediate

Main: <Untitled> Context: Program not running 00000:000

```

٢ - نفذ البرنامج، لاحظ استخدام عبارة REM في البرنامج واضغط على أى مفتاح للعودة الى البرنامج.

```

99.99905

Press any key to continue

```

٣ - من قائمة File اختر New واكتب N لاختلاء الشاشة.

٤ - انتقل الى الدرس المائة والثامن عشر للاستمرار فى تسلسل التعلم.

الدرس المائة والسابع عشر

عبارة RESET

الوصف

تغلق عبارة RESET كل الملفات المفتوحة على القرص. وتكوينها هو كما يلي :

```
RESET
```

ولا تستخدم أى قيم مع عبارة RESET للتمرير كما انها لاتعيد أى قيمة وتكتسب العبارة البيانات فى الذاكرة الاحتياطية النهائية للملفات وتغلق كل الملفات .

التطبيقات

عبارة RESET هى طريقة جيدة لضمان ان ملفات القرص تم تجديدها واغلاقها بطريقة مناسبة قبل فصل البرنامج. وتستخدم كأخر عبارة فى البرنامج قبل فصله . وفيما يلى مثال لها .

```
OPEN "Client.Act" FOR APPEND AS #1
..
PRINT #1, CLRec
..
RESET
END
```

عملية تقليدية

هذه العملية توضح عبارة RESET . ابدأ بتحميل ببسك السريع .

١ - اكتب البرنامج التالى :


```

File Edit View Search Run Debug Calls F1=Help
<Untitled>
This program demonstrates the RESET statement. The program from
Module 19 is modified to use the RESET statement instead of the CLOSE
statement. A new file, PRINTZ.FIL, is created.

CLS
OPEN "PRINTZ.FIL" FOR OUTPUT AS #3
L1$ = "Earth Moving Equipment": L2$ = "12": L3$ = "$120,000"
The PRINT # statement.
L1$ = "Farm Equipment": L2$ = "22": L3$ = "$85,000"
PRINT #3, L1$, L2$, L3$
Format$ = "\ \ \"
PRINT #3, USING Format$: L1$, L2$, L3$
PRINT #3, USING "!": "Save", "Our", "Souls"
RESET

Immediate
Main: <Untitled> Context: Program not running 800009:052

```

- ٢ - نفذ البرنامج. لاحظ استخدام عبارة RESET في البرنامج .
- ٣ - ارجع الى البرنامج واختر New دون ان تحفظ هذا البرنامج.
- ٤ - انتقل الى الدرس المائة والتاسع والثلاثين للاستمرار في تسلسل التعلم.

الدرس المائة والثامن عشر

عبارة RESTORE

الوصف

يمكن عبارة RESTORE برنامج البيسك من اعادة قراءة بيانات من عبارات DATA عندما يتحدد رقم سطر او اسم سطر فتقرأ البيانات منه، وتكونها هو كما يلي :

RESTORE line number or label

عندما ينفذ البرنامج عبارة RESTORE فتقرأ عبارة READ التالية البيانات من سطر محدد فإذا لم يكن هناك رقم او اسم سطر محدد فتقرأ البيانات من اول عبارة DATA موجودة فى البرنامج، وعندما يتحدد رقم او اسم للسطر فيجب ان توجد الاشارة له على مستوى الجزء الرئيسى . وتنقل كل عبارات DATA تلقائيا فى بيسك السريع الى شفرة على مستوى الجزء الرئيسى.

التطبيقات

تستخدم عبارة RESTORE فى قراءة نفس عبارة DATA اكثر من مرة واحدة، وهذا مفيد تحت شروط عديدة . تحميل منظومة هو احد هذه الشروط واعادة وضع قيم ابتدائية لمتغيرات لقيمها الاصلية هى شرط اخر، وفيما يلي بعض الامثلة :

```

READ q1, q2, q3
DATA 24.30.555
RESTORE
READ q1, q2, q3

```

يبين هذا المثال عبارات READ و DATA و RESTORE فى اعداد تقليدى فتقرأ أول عبارة READ القيم q1 و q2 و q3، ويستمر البرنامج فى التنفيذ ليلتقى بعبارة RE-STORE وعند ذلك يسمح بإعادة قراءة قيم q1 و q2 و q3.

```

..READ St1$, St2$, St3$, t1%, t2%, R1!, R2!
..
DATA 1313, Mockingbird Ln., Munsters
Label1:
DATA 11.90
Label2:
DATA 0.33133, 9.78665
..
RESTORE Label1
READ t1%, t2%
..
RESTORE Label2
READ R1!, R2!

```

يوضح هذا المثال استخدام عبارة RESTORE مع اشارة اختيارية للاسم. لاحظ ان اول عبارة RESTORE تشير الى label1 والذي يسمح بالقراءة من عبارة DATA عند هذا الاسم . وتشير عبارة RESTORE الثانية الى label2 ويسمح ذلك بقراءة البيانات من عبارة DATA هذه .

عملية تقليدية

هذه العملية توضح استخدام عبارة RESTORE . ابدأ بتحميل بيك السريع .

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls
Untitled2
This program demonstrates the RESTORE statement

CLS
GOSUB ReadNPrint
GOSUB ReadNPrint
RESTORE
GOSUB ReadNPrint
GOSUB ReadNPrint
RESTORE 11
GOSUB ReadNPrint

END

ReadNPrint:
    READ A$, B$, C$
    PRINT A$, B$, C$
    RETURN

DATA Johnny B. Goode, 1000 BrickWood House, Louisiana
11:
DATA Michael R. B., 120 DeepSea, SouthPacific

```

Immediate

Run: UNTITLED2 Context: Program not running 00000002

٢ - نفذ البرنامج. لاحظ المخرجات وكيفية استخدام البرنامج لعبارات RESTORE في إعادة قراءة البيانات. اضغط على أى مفتاح للعودة الى البرنامج.

Johnny B. Goode	1000 Brick&Wood House	Louisiana
Michael R. B. 120 DeepSea	SouthPacific	
Johnny B. Goode	1000 Brick&Wood House	Louisiana
Michael R. B. 120 DeepSea	SouthPacific	
Michael R. B. 120 DeepSea	SouthPacific	

Press any key to continue

٣ - من قائمة File اختر New واكتب N لا خلاء الشاشة.

٤ - انتقل الى الدرس السابع والأربعين للاستمرار فى تسلسل التعلم.

الدرس المائة والتاسع عشر

عبارة RESUME

الوصف

تتسبب عبارة RESUME فى الاستمرار فى تنفيذ البرنامج بعد اصطيااد احد الاخطاء .
وتكوينها هو كما يلى حيث إنها تأخذ ثلاثة اشكال :

التكوين الاول :

RESUME 0

يجعل هذا التكوين البرنامج يستمر عند السطر الذى حدث عنده الخطأ وجزء 0 من التكوين
يمكن اهماله بدون ان يتغير التأثير.

التكوين الثانى :

RESUME NEXT

يجعل هذا التكوين البرنامج يستمر عند السطر الذى يلى السطر الذى حدث فيه الخطأ
مباشرة.

التكوين الثالث:

RESUME line number[line label]

يجعل هذا التكوين البرنامج يستمر عند رقم سطر او اسم سطر معين. ورقم السطر او اسم
السطر الذى يشار اليه يجب ان يكون على مستوى الجزء. ومن الافضل تجنب هذا التكوين بحيث
يمكن استمرار التنفيذ بغض النظر عن موقع حدوث الخطأ .

عندما تستخدم عبارة RESUME خارج جزء معالجة الخطأ فنتج رسالة بأن عبارة RE-
SUME بدون خطأ . وعندما يحدث خطأ داخل تكوين DEF FN فتستمر عبارة RESUME
وعبارة RESUME NEXT فى تنفيذ البرنامج عند السطر الذى يحتوى على الدالة.

التطبيقات

تستخدم عبارة RESUME بالاتصال مع اجزاء اصطياد الخطأ ومعالجة الخطأ مثل عبارة ON ERROR GOTO. وفيما يلي بعض الامثلة :

مثال ١

```
ON ERROR GOTO 1000
1000 PRINT "Error in line "; ERL
      RESUME NEXT
```

مثال ٢

```
ON ERROR GOTO ErrHandler
ErrHandler:
      RESUME
```

عملية تقليدية

هذه العملية توضح عبارة RESUME . ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :

```
File Edit View Search Run Debug Calls F1=Help
<Untitled>
' This program demonstrates the RESUME statement. The program asks
' for a filename to list, and ON ERROR gives a message and continues to
' ask for a filename.
ON ERROR GOTO EH
CLS
Begin:
INPUT "Enter filename to list: "; FileName$
OPEN FileName$ FOR INPUT AS #1

DO WHILE NOT EOF(1)
    LINE INPUT #1, Line$
    PRINT Line$
LOOP

EH:
PRINT "Error number: "; ERR
RESUME Begin

----- Immediate -----
Main: <Untitled> Context: Program not running 00017:015
```

٢ - نفذ البرنامج . لاحظ استخدام عبارة RESUME في البرنامج.

```
Enter filename to list: ? Music
Error number: 53
Enter filename to list: ? NoName
Error number: 53
Enter filename to list: ?
```

٣ - ارجع الى البرنامج واختر New دون ان تحفظ هذا البرنامج .

٤ - انتقل الى الدرس المائة والسابع عشر للاستمرار في تسلسل التعلم.

الدرس المائة والعشرون

دالة RIGHT\$

الوصف

تعيد دالة RIGHT\$ عدد الرموز المحدد على أقصى يمين تعبير سلسلة وتكوينها هو كما يلي:

RIGHT\$(string expression,num)

الجزء	الوصف
RIGHT\$	كلمة من كلمات بيسك المحجوزة.
string expression	سلسلة مؤشر والتي يعود منها عدد num الرموز الموجودة على أقصى اليمين. ويمكن أن يكون تعبير سلسلة أو متغير أو ثابت.
num	عدد الرموز الموجودة في أقصى اليمين والتي تعود من تعبير السلسلة. فإذا كان num مساوياً لطول تعبير السلسلة فتعيد الدالة RIGHT\$ تعبير السلسلة كله.

التطبيقات

دالة RIGHT\$ هي إحدى وسائل تشغيل السلاسل في بيسك السريع واستخداماتها محدودة بالتخيلات والتطبيقات فقط. وفيما يلي بعض الأمثلة.

مثال ١

```
FullName$ = "George B. Shaw"
PRINT RIGHT$(FullName$,4)
```


مثال ٢

```
Path$ = "C:\BASIC\QB4"
FileName$ = RIGHT$(Path$,3) + "\NewComp.Bas"
PRINT FileName$
```

مثال ٣

```
FullName$ = "George B. Shaw"
PRINT RIGHT$(FullName$,40)
```

مثال ٤

```
FullName$ = "Mark Meslone"
PRINT RIGHT$(FullName$,0)
```

عملية تقليدية

هذه العملية توضح برنامجا بسيطا يستخدم دالة RIGHT\$. ابدأ بتحميل بيك السريع.

١ - اكتب البرنامج التالي :

```
File Edit View Search Run Debug Calls F1-Help
<Untitled>

CLS
Var$ = 'ABRACADABRA'
PRINT : PRINT Var$
FOR Cnt = 1 TO LEN(Var$)
  PRINT TAB(LEN(Var$) - (Cnt - 1)); RIGHT$(Var$, Cnt)
NEXT Cnt

Immediate

Name: <Untitled> Context: Program not running L: 00001:009
```

٢ - نفذ البرنامج ولاحظ استخدام دالة RIGHT\$ فى البرنامج.

```
ABRACADABRA
A
RA
BRA
ABRA
DABRA
ADABRA
CADABRA
ACADABRA
RACADABRA
BRACADABRA
ABRACADABRA

Press any key to continue
```

٣ - اضغط على اى مفتاح للعودة الى البرنامج، اضغط على Alt - F ثم اضغط على مفتاح الادخال واكتب N لإخلاء الشاشة.

٤ - انتقل الى الدرس الثالث والسبعين للاستمرار فى تسلسل التعلم.

الدرس المائة والحادى والعشرون

دالة RND

الوصف

تعيد دالة RND رقما عشوائيا يقع بين 0 و 1. والقيمة التى تعود تكون من النوع فردى الدقة. وتكوينها هو كما يلى :

RND (n)

القيمة التى تعيدها دالة RND تتحدد بواسطة n والتى تكون تعبيراً عددياً . ويصف الجدول التالى العلاقة :

النتيجة	N
بعض الأرقام العشوائية بغض النظر عن قيمة n.	$n < 0$
الرقم العشوائى التالى فى التسلسل.	$n > 0$
آخر رقم عشوائى تم انتاجه.	أو محذوفه $n = 0$

فإذا لم توضع قيمة ابتدائية لمنتج الأرقام العشوائية باستخدام دالة RANDOMIZE فيتم انتاج نفس الأرقام حتى اذا ما كان التعبير العددي أكبر من 0.

التطبيقات

دالة RND هى دالة مريحة عندما تكون هناك حاجة الى انتاج سلسلة من الأرقام العشوائية اثناء تنفيذ البرنامج . كمثال لنل هذا الموقف عملية رسم النجوم فى إحدى المباريات المرئية على الشاشة. وفيما يلى بعض الامثلة.

```
T1 = RND(3)
NLoc% = INT((32-10+1)*RND+10)
```

يستنتج هذا المثال رقما عشوائيا صحيحا يقع بين 32 و 10 وذلك من قيمة لها دقة فردية تعيدها دالة RND . وهذه العملية توضح بصورة افضل في المثال التالي .

Integer Random Number = INT ((Upper limit - Lower limit + 1)
* RND + Lower limit)

عملية تقليدية

هذه العملية توضح استخدام دالة RND . ابدأ بتحميل بيسك السريع .

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls F1-Help
<Untitled>
' This program generates random numbers using the RND function.
CLS
PRINT "Numbers generated using the RND function:"
FOR Cnt = 1 TO 20
    PRINT RND(Cnt \ 1.1),
NEXT
Immediate
Main: <Untitled> Context: Properties window VB6000.DLL

```

٢ - نفذ البرنامج ولاحظ استخدام دالة RND في البرنامج. لاحظ ان المؤثر \ يستخدم بدلا من / في قسمة Cnt للحصول على الجزء الكسرى. وحيث إننا لم نضع قيما ابتدائية لمنتج الارقام العشوائية بعبارة RND فيجب ان ينتج البرنامج نفس تسلسل الارقام العشوائية . حاول ان تنفذ البرنامج مرة اخرى وتأكد مما اذا كان ذلك صحيحا ام لا .

```

Numbers generated using the RND function:
.7107346      .99058      .0523988      .3503776      4.363585E-02
0.977669E-02 .5111076      .7553547      .935394
.3444337      .2073416      .5360375      .2351913      .7695047
.2087399      .9520085      .9767802      .8262324      .4440745
.2794519

Press any key to continue

```

٣ - اضغط على اى مفتاح للعودة الى البرنامج . من قائمة File اختر New واكتب N لإخلاء الشاشة.

٤ - انتقل الى الدرس المائة والثالث عشر للاستمرار فى تسلسل التعلم.

الدرس المائة والثاني والعشرون

عبارة RUN

الوصف

تستخدم عبارة RUN فى تنفيذ برامج اخرى من داخل احد برامج بيسك السريع وتكونها
هو كما يلى :

```
RUN line|file spec
```

القضيب الرئيسى (ا) الموجود بين line و filespec يحدد أن أحد هذين المؤشرين فقط هو المستخدم . جزء line هو رقم السطر الذى يبدأ عنده تنفيذ البرنامج . وجزء filespec هو اسم ملف البرنامج المراد تنفيذه . وينفذ البرنامج كما لو كان يبدأ التنفيذ من البداية : كل الملفات مغلقة وقيم المتغيرات الابتدائية يعاد وضعها . ولا يمكنك أن تمرر معلومات الى برامج اخرى او اقتسام متغيرات .

التطبيقات

تستخدم عبارة RUN فى تنفيذ برامج اخرى من برنامج ينفذ حاليا وفى إعادة بدء تنفيذ البرنامج الحالى من عند رقم سطر معين . وفيما يلى امثلة لعبارة RUN .

مثال ١

```
PRINT "Please wait ..."  
RUN "Screen"
```

مثال ٢

```
10 Wrt = 23  
20 PRINT FNDBlWrt(Wrt)  
30 Wrt = Wrt * 2  
40 PRINT FNDBlWrt(Wrt)  
50 RUN 30
```

فى هذا المثال تكون قيمة Wrt مساوية 0 وتطبع عبارة PRINT هذه القيمة لأن التحديد السابق لـ Wrt لا يكون له اى تأثير .

عملية تقليدية

هذه العملية توضح استخدام وتأثير عبارة RUN كما هي محددة في المثال التالي في قسم التطبيقات السابق. ابدأ بتحميل ببسك السريع.

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls F1=Help
<Untitled>
' This program demonstrates the RUN statement.
DEF FnDbl (i AS INTEGER) = i * 2

CLS
Wrt = 3
PRINT FnDbl(Wrt), Wrt
10 Wrt = Wrt * 2
PRINT FnDbl(Wrt), Wrt
IF Wrt = 1 THEN END
IF Wrt > 8 THEN RUN 10

```

Immediate

Main: <Untitled> Context: Program not running 00010-022

٢ - نفذ البرنامج. لاحظ المخرجات التالية وتأثير عبارة RUN على قيم المتغير.

```

6      3
12     6
0      0

```

Press any key to continue

٣ - ارجع الى البرنامج مع اخلاء الشاشة دون ان تحفظه.

٤ - انتقل الى الدرس المائة والرابع والعشرين للاستمرار فى تسلسل التعلم.

الدرس المائة والثالث والعشرون

دالة SADD

الوصف

تعطى دالة SADD عنوان تعبير سلسلة محدد، وتكوينها هو كما يلي :

```
SADD(string exp)
```

جزء string exp يكون متغير سلسلة او عنصراً من عناصر منظومة سلاسل، وغير مسموح بسلاسل ثابتة الطول، والعنوان الذى يعود هو فرع من قطاع البيانات الحالى. ولا يعمل استخدام SADD فى قطاعات غير قطاع بيانات بيسك السريع.

التطبيقات

تستخدم دالة SADD فى معظم الاحيان فى برمجة بخليط من اللغات حيث تكتب الاجزاء بلغات غير لغة بيسك السريع. وحيث انه يمكن ان تتحرك السلسلة داخل الذاكرة اثناء تنفيذ البرنامج فيجب ان تستخدم دالة SADD بحذر. اضافة رموز الى تعبير سلسلة المؤشر يقود الى حدوث خطأ وقت التنفيذ ، وفيما يلى مثال لدالة SADD.

```
T$ = "ABCDEFGH"  
PRINT SADD(T$)
```

عملية تقليدية

حيث إننا لا نستطيع ان نفترض ان القارئ لديه امكانية اتصال بمرجم لغة C من ميكروسوفت أو انه لديه معرفة بالبرمجة بلغة C فإن المثال الموجود فى هذا القسم محدود بتقديمه توضيحاً بسيطاً لنوع القيمة التى تعود بواسطة دالة SADD ، ابدا بتحميل بيسك السريع .

١ - اكتب البرنامج التالى :

```

File Edit View Search Run Debug Calls
<Untitled>
' This program demonstrates the SADD function.
CLS
T$ = "Example string"
PRINT SADD(T$)

Immediate

Main: <Untitled> Context: Program not running 00004:015

```

٢ - نفذ البرنامج. تكون المخرجات على النحو التالي :

```

13950

Press any key to continue

```

٣ - ارجع الى البرنامج مع اخلاء الشاشة لى ان تحفظ هذا البرنامج.

٤ - انتقل الى الدرس المائة والسادس والخمسين للاستمرار فى تسلسل التعلم.

الدرس المائة والرابع والعشرون

دالة وعبارة SCREEN

الوصف

تستخدم الكلمة المحجوزة SCREEN كدالة او كعبارة . وعندما تستخدم كدالة فإنها تعيد رمزا عند احداثيات الشاشة المعطاة. وعندما تستخدم كعبارة فإنها تعرف كيفية اعداد الكمبيوتر. وكل من الاستخدامين مذكور في هذا الدرس .

دالة SCREEN: تكوينها هو كما يلي :

SCREEN (row, col, color flag)

جزءاً row و col هما احداثيات الصف والعمود وجزء color flag هو مؤشر اختياري. تعيد SCREEN اللون عند الاحداثيات المستخدم عندها color flag وتعيد رمز قيمة ASCII عند الاحداثيات غير المستخدم عندها color flag . وتقوم كل المؤشرات بقيم عددية صحيحة وتكون الاقواس ضرورية حول المؤشرات.

عبارة SCREEN: تحدد عبارة SCREEN الخواص الخاصة بعرض الكمبيوتر. وتستخدم العبارة في اختيار حالة العرض المناسبة لبرنامج تطبيق معين ونظام كمبيوتر معين. وتكوينها هو كما يلي :

SCREEN mode, color switch, active page, visual page

جزء mode عبارة عن تعبير عددي صحيح يصف حالة الشاشة . والجدول التالي يوضح القيم المختلفة للحالات وتأثيراتها المصاحبة لها ومتطلباتها .

الحالة	التأثيرات والمتطلبات
0	هذه هي الحالة النصية المعتادة (25 x 40 او 25 x 80 او 43 x 40 او 50 x 40 او 43 x 80 او 50 x 80) وحجم الرمز 8x8 (او 14 x 8 او 14 x 9 او 16 x 9 نقطة رسم مع بطاقة EGA او بطاقة VGA). وتسمح حتى 16 لونا و 2 خاصة او 16 لونا و 16 خاصة مع بطاقة EGA.
1	رسومات متوسطة الثبات (200 x 320 نقطة رسم) وعرض نصوص 40x25 و 16 لوناً

الحالة	التاثيرات والمتطلبات
2	و 4 خواص مع بطاقة EGA وتدعم بطاقات CGA و EGA و VGA و MCGA. رسومات مرتفعة الثبات (640 x 200) وعرض نصوص 80 x 25 و 16 لوناً و 2 خاصية مع بطاقة EGA او بطاقة VGA ، وتدعم بطاقات CGA و EGA و VGA و MCGA.
7	رسومات متوسطة الثبات (320 x 200) وعرض نصوص 40 x 25 وصفحات متعددة الشاشات و 16 لوناً مع 16 خاصية وتتطلب بطاقة EGA او بطاقة VGA.
8	رسومات مرتفعة الثبات (640 x 200) وعرض نصوص 80 x 25 وصفحات متعددة الشاشات و 16 لوناً مع 16 خاصية وتتطلب بطاقة EGA او بطاقة VGA.
9	رسومات معززة (640 x 300) وعرض نصوص 80 x 25 او 80 x 43 برموز 8 x 8 او 8 x 14 و 64 لوناً مع 16 خاصية او 16 لوناً مع 4 خواص طبقاً لما اذا كانت الذاكرة بها بطاقة EGA او VGA وصفحات متعددة الشاشات وتتطلب EGA او VGA.
10	مثل الحالة 9 وتسمح بعدد 9 ألوان شبيهة مع 4 خواص .
11	رسومات مرتفعة الثبات جدا (640 x 480) وعرض نصوص 80 x 30 او 80x60 برموز 8x8 او 8x16 و 256K لوناً و 2 خاصية وتتطلب EGA او VGA.
12	مثل الحالة 11 وتسمح ب 256K لوناً من 16 خاصية وتتطلب VGA.
13	رسومات متوسطة الثبات (320x200) وعرض نصوص 40x25 و 256K لوناً و 256 خاصية وتتطلب VGA او MCGA.

جزء color switch فى تكوين عبارة SCREEN يحدد ما اذا كان اللون معروضا ام لا على
شاشات مركبة. و color switch هو قيمة تقع فى المدى من 0 الى 255 وعندما لا تكون القيمة
صفراً فتعرض صوراً ابيض واسود فقط ويعرض اللون عندما تكون صفراً. وفى حالة الشاشة 0
فتتحول القيمة، وتهمل القيمة فى حالات الشاشة من 2 واكثر.

جزء active page هو صفحة الشاشة التى تكتب فيها عبارات الرسومات، وجزء visual
page هو الجزء المعروض حالياً.

وعند استخدام حالة الشاشة 0 مع عرض IBM احادى اللون وضابط طابع فيكون التأثير كما يلى :

Mode	0
RowsxCol	25x80
Attributes	16
Colors	3
Resolution	720x350
Pages	1

ومع ضابط الرسومات الملونة CGA من طراز IBM فما يلى هو حالات الشاشة وتأثيراتها.

Mode	RowsxCols	Colors	Resolution	Pages
0	40x25,80x25	16,16	320x200, 640x200	8 4
1	40x25	4	320x200	1
2	80x25	2	640x200	1

ومع ضابط رسومات معرزة EGA تكون حالات الشاشة وتأثيراتها كما يلى :

Mode	RowsxCols	Display	Attr.	Colors	Res.	Page,Page size
0	40x25	C	16	16	320x200	8,NA
	40x25	E	"	64	320x350	"
	40x43	"	"	"	"	"
	80x25	C	"	16	640x200	"
	80x25	E	"	64	640x350	"
	80x25	C	"	16	640x200	"
	80x25	M	"	3	720x350	"
	80x43	E	"	64	640x350	4,NA
	80x43	M	16	3	720x350	"
1	40x25	NA	4	16	320x200	1,16K
2	80x25	"	2	"	640x200	"
7	40x25	"	16	"	320x200	1,32K
8	80x25	"	"	"	640x200	1,64K
9	80x25	E	4	64	640x350	"
	80x43	"	"	"	"	"
	80x25	"	16	"	"	1,128K
	80x43	"	"	"	"	"
10	80x25	M	4	9	"	1,64K
	80x43	"	"	"	"	"

ومع منظومة رسومات مرئية VGA تكون حالات الشاشة وتأثيراتها كما يلي :

Mode	RowsxCols	Attr.	Colors	Res.	Page, Page size
0	40x25	16	64	360x400	8, NA
	40x43	"	"	320x350	"
	40x50	"	"	320x400	4, NA
	80x25	"	"	720x400	8, NA
	80x43	"	"	640x350	4, NA
	80x43	"	3	720x350	"
	80x50	"	64	640x400	"
	80x50	"	3	720x400	"
1	40x25	4	16	320x200	1, 16K
2	80x25	2	"	640x200	"
7	40x25	16	"	320x200	1, 32K
8	80x25	"	"	640x200	1, 64K
9	80x25	"	64	640x350	1, 128K
	80x43	"	"	"	"
10	80x25	4	9	"	1, 64K
	80x43	"	"	"	"
11	80x30	2	256K	640x480	"
	80x60	"	"	"	"
12	80x30	16	"	"	1, 256K
	80x60	"	"	"	"
13	40x25	256	"	320x200	1, 64K

ومع منظومة رسومات متعددة الالوان MCGA تكون حالات الشاشة وتأثيراتها كما يلي :

Mode	RowsxCols	Attr.	Colors	Res.	Page, Page size
0	40x25	16	NA	320x400	8, NA
	80x25	"	"	640x400	"
1	40x25	4	"	320x200	1, 16K
2	80x25	2	"	640x200	"
11	80x30	"	256K	640x480	1, 64K
	80x60	"	"	"	"
13	40x25	256	"	320x200	"

التطبيقات

تستخدم عبارة SCREEN في تحديد حالة العرض المستخدمة في برنامج تطبيق معين. ويجب ان يتم اختيار الحالة بعناية ويعتمد ذلك على نظم المكونات المتاحة . واختيار الحالة الصحيحة للشاشة للعمل فيها يكون مهما عندما يكون البرنامج متداخلا جدا او ينتج ويستخدم نوافذ او يستخدم رسومات. ويجب اختيار الحالة المناسبة للشاشة عند استخدام عبارات رسومات مثل DRAW و CIRCLE. وفيما يلي بعض الامثلة لعبارة SCREEN:

مثال ١

```
SCREEN 2
..
```

مثال ٢

```
SCREEN 1: COLOR 1
LINE (0,0)-(319-199)
..
```

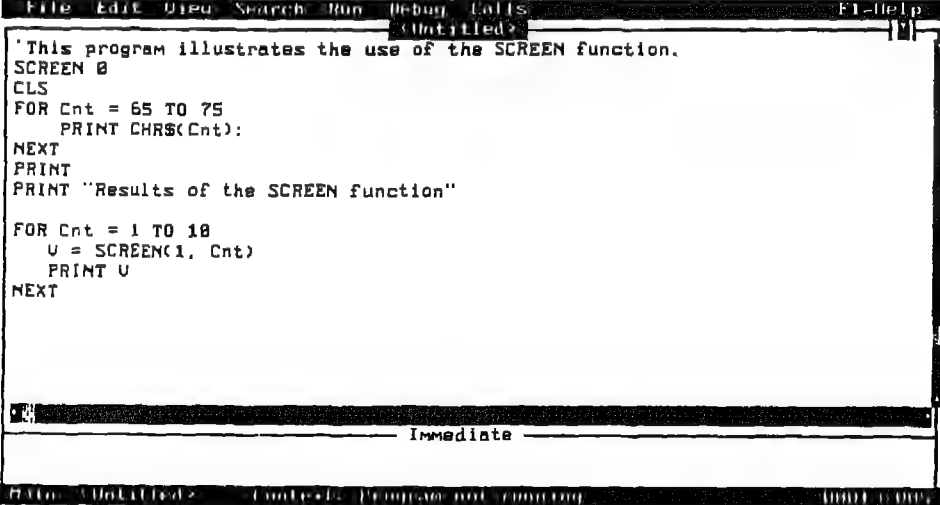
مثال ٣

```
SCREEN 2
DRAW D$
```

عملية تقليدية

توجد امثلة لعبارة SCREEN في الدروس 76 و 35 و 16 و 97. وهذه العملية توضح دالة SCREEN في صورة مبسطة. ابدأ بتحميل بيك السريع .

١ - اكتب البرنامج التالي :



```
File Edit View Search Run Debug Calls F1-Help
Untitled
' This program illustrates the use of the SCREEN function.
SCREEN 0
CLS
FOR Cnt = 65 TO 75
    PRINT CHR$(Cnt):
NEXT
PRINT
PRINT "Results of the SCREEN function"

FOR Cnt = 1 TO 10
    U = SCREEN(1, Cnt)
    PRINT U
NEXT

Immediate
Status: Untitled - Controls: Program not running 100010000
```

٢ - نفذ البرنامج، لاحظ استخدام دالة SCREEN وعبارة SCREEN في البرنامج، وتشبه المخرجات ما يلي :

```

ABCDEFGHIJK
Results of the SCREEN function
65
66
67
68
69
70
71
72
73
74

Press any key to continue

```

٣ - ارجع الى البرنامج مع اخلاء الشاشة دون ان تحفظ هذا البرنامج.
٤ - انتقل الى الدرس السادس والسبعين للاستمرار في تسلسل التعلم.

الدرس المائة والخامس والعشرون

عبارة دالة و SEEK

الوصف

تستخدم SEEK كدالة و كعبارة . وتضع عبارة SEEK مشير الملف عند الموقع المحدد. وتعطى دالة SEEK موقع مشير الملف داخل الملف. وتهمل عبارات و دوال SEEK مع الوحدات التالية : SCRN و CONS و : KYBRD و COMx و LPTx حيث x رقم صحيح.

عبارة SEEK : وتكوينها هو كما يلي :

SEEK #filenum, pos

جزء filenum هو رقم الملف المحدد فى عبارة OPEN . جزء pos هو الموقع الذى ينقل اليه مشير الملف. وهذا هو المكان الذى تبدأ فيه عملية القراءة او عملية الكتابة التالية. ويمكن ان تصل قيمة pos حتى 2,147,483,647 كحد اقصى. ومع ملفات الاتصال العشوائى تكون pos هى رقم السجل . ومع الملفات المفتوحة بأنها ملفات BINARY او INPUT و OUTPUT او APPEND يكون pos هو موقع البايت فى الملف ودائما ما يكون pos اكبر من صفر.

دالة SEEK : وتكوينها هو كما يلي :

SEEK(filenum)

جزء filenum هو نفسه مثل ما هو موجود فى عبارة SEEK . والقيمة التى تعود من دالة SEEK تقع بين 1 و 2, 147, 483, 647. وعند استخدامها مع ملفات اتصال عشوائى فإن القيمة التى تعود هى موقع السجل التالى . ومع الملفات المفتوحة على انها BINARY او IN- PUT او OUTPUT او APPEND فإن القيمة التى تعود تكون موقع البايت التالى .

التطبيقات

تستخدم عبارة SEEK فى القفز هنا وهناك داخل الملف اثناء تشغيله. وعبارات SEEK المحسوبة هى ميزة للاتصال بالسجلات الموجودة فى الملف. ودالة SEEK لها نفس العمل فى الاساس مثل دالة LOC, وفيما يلي بعض الامثلة :

```
OPEN "Inven.Dat" FOR BINARY AS #1
..
SEEK #1,256
OPEN "Address.Dat" FOR RANDOM AS #1 LEN = 200
..
SEEK #1,10
```

فى المثال السابق تنقل عبارة SEEK المشير الى السجل العاشر. وفى المثال الذى يسبقه تنقل عبارة SEEK المشير الى موضع البايت 256 . وفيما يلى امثلة لدالة SEEK :

```
SEEK #3, SEEK(3) - (LEN(RecordLen) * 3)
```

المثال السابق يستخدم دالة SEEK فى عمل عبارة SEEK محسوبة. ينتقل مشير السجل الى الخلف بثلاثة سجلات.

عملية تقليدية

هذه العملية توضح عبارة SEEK . ابدأ بتحميل بيسك السريع .

١ - اكتب البرنامج التالى :

```
File Edit View Search Run Debug Calls F1=Help
<Untitled>
'This program demonstrates the SEEK statement. The program uses SEEK
'to read in a particular record from the file created in Module 74.

TYPE CustType
    CustName AS STRING * 25
    CustNum AS INTEGER
    CustType AS STRING * 2
END TYPE

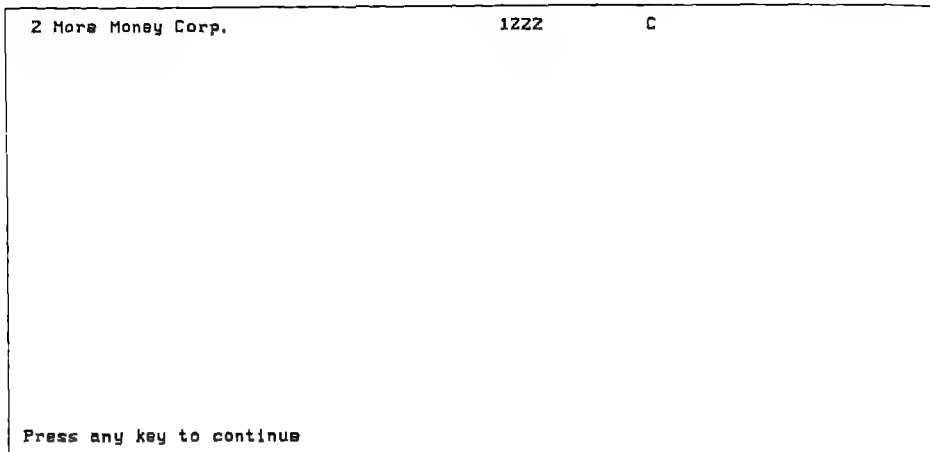
DIM Customer AS CustType

OPEN "Cust.Fil" FOR RANDOM AS #2
SEEK #2, 2
GET #2, , Customer
PRINT LOC(2); Customer.CustName, Customer.CustNum, Customer.CustType

CLOSE #2

----- Immediate -----
Main: <Untitled> Context: Program not running 00017:009
```

٢ - نفذ البرنامج ، لاحظ استخدام عبارة SEEK فى البرنامج.



٣ - اضغط على اى مفتاح للعودة الى البرنامج.

٤ - اختر New من قائمة File واختر عدم حفظ هذا البرنامج.

٥ - انتقل الى الدرس الحادى والثمانين للاستمرار فى تسلسل التعلم.

الدرس المائة وستة وعشرون

عبارة SELECT CASE

الوصف

عبارة SELECT CASE هي طريقة اكثر ترتيبا فى كتابة عبارة IF THEN ELSE متعددة المستويات، وتكوين عبارة SELECT CASE هو كما يلى :

```
SELECT CASE test exp
CASE test1
statements
CASE test2
statements
CASE ELSE
statements
END SELECT
```

وفيما يلى وصف اجزاء التكوين:

الوصف	الجزء
هذا هو التعبير الذى يتم تقويمه لتحديد اى فرع تتفرع اليه عبارة SELECT CASE لتنفيذه .	test exp
هذه هي النتائج الممكنة لتعبير الاختبار test exp .	test1, test 2
عبارات بيسك السريع التى تنفذ عندما تكون نتيجة test exp صحيحة بالنسبة الى test 1 او test 2 .	statements

test1 و test2 لهما الصيغ التالية :

```
CASE test1, test2,...
CASE test1 TO test2
CASE test1 IS (<,>,<=,>=,<>,<=,>=)
```

اول صيغة هي قائمة بالنتائج الممكنة لتقويم التعبير المختبر. وتنفذ مجموعة العبارة عندما تكون النتيجة الفعلية واحدة من القائمة . وتعطى الصيغة الثانية مدى لقيم من test1 الى test2 وتنفذ مجموعة العبارة عندما تقع النتيجة داخل المدى . والصيغة الثالثة علاقية فى طبيعتها . ويمكن استخدام اى من المؤثرات العلاقية الموجودة بين الاقواس والاقواس نفسها غير مشمولة. وتنفذ مجموعة العبارة هذه عندما تقوم العملية العلاقية بأن قيمتها صحيحة TEUE .

ويقوم جزء CASE ELSE فى التكوين للتعامل مع النتائج الممكنة الاخرى غير الموجودة فى عبارات CASE السابقة . وتنفذ مجموعة العبارة هذه عندما لا يتحقق اى من الاختبارات التالية اى عندما تكون نتيجتها كلها FALSE . وجزء CASE ELSE اختياري .

وتكوين SELECT CASE مثير عندما يتحقق شرط واحد وتنفذ مجموعة العبارة. وهذا هو سبب ان عبارة SELECT CASE عبارة عن طريقة اكثر ترتيبا فى كتابة عبارات IF THEN ELSE متعددة.

التطبيقات

عبارة SELECT CASE وسيلة مفيدة فى البرمجة المرتبة التى تقود الى برامج متماسكة وسهلة القراءة وتكون سهلة كذلك فى صيانتها. وفيما يلى امثلة لعبارات SELECT CASE .

```
SELECT CASE Choice$
CASE IS "A"
GOSUB AddRecords
CASE "E" TO "H"
GOSUB EditProces
CASE ELSE
GOSUB SurpriseProcess
END SELECT

INPUT "Enter value: ";V%
SELECT CASE V%
CASE IS > 10
..
CASE i TO 3
..
END SELECT
```

عملية تقليدية

فى هذه العملية تقوم بتعديل البرنامج المقدم فى الدرس الثامن والخمسين لاستخدام عبارة SELECT CASE . ابدأ بتحميل ببسك السريع.

١ - من قائمة File اختر Open واضغط على Tab للذهاب الى الدليل . اختر BAS THEN IF واضغط على مفتاح الادخال.

٢ - عدل عبارة IF THEN ELSE كما هو مبين فى القائمة التالية :

```

File Edit View Search Run Debug Calls FI=Help
CASE.BAS
CLS
PRINT "This is a guessing game."
PRINT "I have a number in mind. You try to guess what it is."
RANDOMIZE TIMER
Guess% = INT(10 - 1) * RND + 1
PRINT "You have three guesses..."
Cnt = 1
DO WHILE (Cnt <= 3)
    INPUT "Your guess ", InG%
    SELECT CASE InG%
        CASE IS > Guess%
            PRINT "Too high.."
        CASE IS < Guess%
            PRINT "Too low.."
        CASE IS = Guess%
            PRINT "Bingo !"
            Cnt = 4
    END SELECT
    Cnt = Cnt + 1
LOOP
PRINT "My number was "; Guess%; " and your number was "; InG%

```

Immediate

Main: CASE.BAS Context: Program not running 000-Jb/001

٣ - نفذ البرنامج وادخل تخميناتك.

```

This is a guessing game.
I have a number in mind. You try to guess what it is.
You have three guesses...
Your guess 3
Too low..
Your guess 5
Too low..
Your guess 12
Too high..
My number was 8 and your number was 12

```

Press any key to continue

٤ - لاحظ استخدام عبارة SELECT CASE فى البرنامج وكيفية تحسينه فى قراءة البرنامج.
اضغط على اى مفتاح للعودة الى قائمة البرنامج.

٥ - سوف نحفظ هذا البرنامج فى صورته المعدلة مستخدما اسم ملف مختلف من قائمة File
اختر Save As واحذف اسم الملف الاصلى بالضغط على Del ثم اكتب CASE.BAS
كاسم للملف، حدد أن شكل الملف نصى واحفظ هذا البرنامج.

٦ - من قائمة File اختر New مع اخلاء الشاشة .

٧ - انتقل الى الدرس التاسع عشر للاستمرار فى تسلسل التعلم.

الدرس المائة والسابع والعشرون

دالة SETMEM

الوصف

دالة SETMEM تزيد او تقلل من الذاكرة المستخدمة بواسطة الكومة البعيدة، والكومة البعيدة far heap هي المكان المخزن فيه الاشياء البعيدة وتكوين العبارة هو كما يلي :

SETMEM(number)

جزء number هو تعبير عددي يمكن ان يكون سالبا او موجبا . وعندما يكون موجبا تزداد الكومة البعيدة بعدد البايت المحدد، وعندما يكون العدد اكبر من الذاكرة المتاحة فتتحدد كل الذاكرة المتاحة، وتعيد الدالة كمية ذاكرة الكومة البعيدة المحددة بالبايت، وعندما يكون العدد سالبا فتقل الذاكرة بعدد البايت المحدد، فاذا كان العدد صفرا فتعود دالة SETMEM بالحجم الحالي للكومة البعيدة.

التطبيقات

تستخدم دالة SETMEM في معظم الحالات في البرمجة بخليط من اللغات، وفيما يلي مثال لدالة SETMEM .

```
DIM R(100)
Fh = SETMEM(0)
PRINT "Far heap = ";Fh;"bytes"
```

عملية تقليدية

العملية التالية توضح دالة SETMEM في صورة بسيطة . والبرنامج عبارة عن تنفيذ للمثال الموجود في قسم التطبيقات، ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :


```

File Edit View Search Run Debug Calls F1-Help
<Untitled>
' This program demonstrates the SETMEM Function.
CLS
DIM T(100) AS STRING
' get the current size of the far heap
Fh = SETMEM(0)
PRINT "Far heap = "; Fh; " bytes"

----- Immediate -----
Main: <Untitled> Context: Program not running 00000000

```

٢ - نفذ البرنامج. لاحظ نتيجة دالة SETMEM في البرنامج.

```

Far heap = 346352 bytes

Press any key to continue

```

٣ - ارجع الى البرنامج مع اخلاء الشاشة بون ان تحفظ هذا البرنامج.

٤ - انتقل الى الدرس المائة والخمسين للاستمرار في تسلسل التعلم.

الدرس المائة والثامن والعشرون

دالة SGN

الوصف

دالة SGN تعيد اشارة التعبير العددي، وتكوينها هو كما يلي :

`SGN(numeric expression)`

نعطى دالة SGN احدى ثلاث نتائج ممكنة طبقا للتعبير العددي، والجدول التالي يسرد النتائج الممكنة .

النتيجة	السبب
1	التعبير العددي اكبر من 0 .
- 1	التعبير العددي اقل من 0 .
0	التعبير العددي مساويا 0 .

التطبيقات

دالة SGN تكون مفيدة عندما تكون هناك حاجة الى اشارة العدد لاتخاذ قرار معين فى البرنامج، (يمكن الحصول على نفس التأثير باختبار ما إذا كانت القيمة اكبر من او مساو لـ او اقل من صفر)، وفيما يلي امثلة تستخدم دالة SGN .

```
PRINT SGN(12-23)
IF SGN(BalanceDue!) = 1 THEN GOTO CreditRecovery
PRINT SGN((2.1/3)-Srt%)
```

عملية تقليدية

توضح دالة SGN فى هذه العملية التقليدية . ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls F1=Help
(Untitled)
The program demonstrates the SGN function
x = 42: y = 10: z = -5
a = x - (y * z)
b = (y + z) * 0
c = (x / y) * z
PRINT a, "SGN(A) = "; SGN(a)
PRINT b, "SGN(B) = "; SGN(b)
PRINT c, "SGN(C) = "; SGN(c)

Immediate

Main: (Untitled) Context: Program not running 00000:029

```

٢ - نفذ البرنامج ولاحظ استخدام دالة SGN في البرنامج. اضغط على أى مفتاح للعودة الى البرنامج.

```

. 92          SGN(A) = 1
0            SGN(B) = 0
-21          SGN(C) = -1

Press any key to continue

```

٣ - من قائمة File اختر New واكتب N لإخلاء الشاشة.

٤ - انتقل الى الدرس المائة والرابع والخمسين للاستمرار فى تسلسل التعلم.

الدرس المائة والتاسع والعشرون

عبارة SHARED

الوصف

تقدم عبارة SHARED اتصالاً بمتغيرات سبق توضيحها على مستوى الجزء إلى SUB و FUNCTION وتكوينها هو كما يلي :

```
SHARED var1 AS type, var2 AS type,...
```

اجزاء var1 و var2 هي اسماء متغيرات وهي متغيرات بيسك سريع صحيحة وتشمل المنظومات كذلك. جزء AS type يحدد نوع بيانات المتغير. يمكن ان يكون النوع من انواع البيانات البسيطة او الانواع التي يحددها المستفيد. وعندما يكون المتغير منظومة فيتبعه قوسان فارغان مثلما يلي :

```
var1(), var2(), ..
```

التطبيقات

يمكن ان تظهر عبارة SHARED داخل SUB او FUNCTION فقط ويمكن ان تقتسم متغيرات موضحة في هذا الجزء فقط وليس مع مكتبة سريعة او اى جزء اخر. يسمح ذلك لـ SUB او FUNCTION ان يستخدم هذه المتغيرات دون ان تمرر كمؤشرات. وفيما يلي امثلة لعبارة SHARED .

مثال ١

```
SUB Fewer
  SHARED Old AS INTEGER, New AS INTEGER
  ..
END SUB
```

مثال ٢

```
FUNCTION ConstStr$
  SHARED Char AS STRING
  ..
END FUNCTION
```

عملية تقليدية

هذه العملية توضح استخدام عبارة SHARED . ابدأ بتحميل ببسك السريع.

١ - حمل برنامج SUB.BAS الذى سبق اعداده فى الدرس المائة والرابع والأربعين وعدل البرنامج كما هو مبين فى القائمة التالية :

The image shows two screenshots of a BASIC IDE. The top screenshot shows the main program SUB.BAS, and the bottom screenshot shows the SUB SortArray subroutine.

```

File Edit View Search Run Debug Calls F1=Help
SUB.BAS
DECLARE SUB SortArray ()
'This program illustrates the use of the SUB statement. The program
'sorts an array in ascending order.

Max = 15
DIM A%(Max)
FOR Cnt = 1 TO Max
  READ A%(Cnt)
NEXT

CALL SortArray
CLS : PRINT
PRINT "The sorted array:"

FOR Cnt = 1 TO Max
  PRINT A%(Cnt);
NEXT

DATA 12,23,789,98,545,22,1,87,43,53,52,333,24,67,88

File Edit View Search Run Debug Calls F1=Help
SUB.BAS: SortArray
SUB SortArray
  SHARED A%()
  FOR C1 = LBOUND(A%) TO UBOUND(A%) - 1
    FOR C2 = C1 + 1 TO UBOUND(A%)
      IF A%(C1) > A%(C2) THEN
        Temp = A%(C2)
        A%(C2) = A%(C1)
        A%(C1) = Temp
      END IF
    NEXT C2
  NEXT C1
END SUB

Immediate

Main: SUB.BAS Context: Program not running 00002:001
  
```

٢ - نفذ البرنامج . لاحظ استخدام عبارة SHARED فى جعل المنظومة متاحة لـ SUB .
مخرجات البرنامج هى كما يلى :

The sorted array:

1 12 22 23 24 43 52 53 67 87 88 90 333 545 789

Press any key to continue

٣ - ارجع الى البرنامج مع اخلاء الشاشة دون ان تحفظ هذا البرنامج.

٤ - انتقل الى الدرس المائة والثلاثين للاستمرار في تسلسل التعلم.

الدرس المائة والثلاثون

عبارة SHELL

الوصف

تسمح لك عبارة SHELL بالخروج من البرنامج وتنفيذ امر DOS والعودة مرة اخرى الى البرنامج. وتكوينها هو كما يلي :

SHELL command

جزء command هو امر DOS صحيح مثل DIR او COPY ويوضع بين علامتى تنصيص. وهو جزء اختياري وعندما تستخدم عبارة SHELL بدونه فإنها تأخذك الى DOS وتسمح لك بتنفيذ اى امر من اوامر DOS والعودة بعد ذلك الى البرنامج عندما تكتب "EXIT". ومن الممكن تنفيذ برامج باستخدام هذه الوسيلة.

التطبيقات

عبارة SHELL مفيدة فى تنفيذ ملفات EXE و .COM و .BAT من البرنامج وفى تنفيذ أنشطة DOS اخرى. وفيما يلي امثلة لعبارة SHELL :

مثال ١

SHELL "DIR /W"

مثال ٢

SHELL "TYPE TOC | SORT > TOC.SRT"

مثال ٣

SHELL

يخرج المثال الاول الى DOS ويعرض الملفات الموجودة فى الدليل المفتوح ويعود الى البرنامج. وينسخ المثال الثانى ملف TOC ويستخدم مرشح ترتيب DOS ويعيد ترتيب المخرجات الى ملف TOC.SRT آخر ثم يعيدك الى البرنامج. ويخرج المثال الثالث من DOS وينتظر امرا. يعود التحكم الى البرنامج عندما تكتب EXIT.

عملية تقليدية

هذا هو توضيح بسيط لعبارة SHELL. استمر اذا كان لديك طابع متصل بالكمبيوتر فقط.
 ابدأ بتحميل بيسك السريع.
 ١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls F1-Help
<Untitled>
'This program demonstrates the SHELL statement.
CLS
PRINT "This program uses the printer. The logged directory is listed "
PRINT "on the printer."
SHELL "Dir / w > Lpt1:"

Immediate

Main: <Untitled> Context: Program not running 000006:001
    
```

٢ - نفذ البرنامج . عند التنفيذ تشبه الشاشة ما يلي : لاحظ استخدام عبارة SHELL في البرنامج.

```

This program uses the printer. The logged directory is listed
on the printer.

Press any key to continue
    
```


٣ - ارجع الى البرنامج اخرج من بيسك السريع دون ان تحفظ البرنامج. بهذا تكون قد اتممت تسلسل التعلم. استمر في الملاحق للمزيد من المعلومات.

الدرس المائة واحد وثلاثون

دالة SIN

الوصف

تحتسب دالة SIN جيب الزاوية المعطاة بالتقدير الدائري. وتكونها هو كما يلي :

`SIN(numeric expression)`

قيمة SIN تحتسب بدقة فردية كقيمة تقليدية، وعندما يكون التعبير العددي بدقة مزدوجة فتحتسب قيمة SIN بدقة مزدوجة. ويمكن ان يكون التعبير العددي اى عدد.

التطبيقات

تستخدم دالة SIN عندما يراد حساب جيب الزاوية كما فى حالة رسم الرسومات. وفيما يلي بعض امثلة لدالة SIN.

```
PRINT SIN(12.22)
T = SIN(.03)
PRINT T
```

عملية تقليدية

العملية التالية توضح دالة SIN ابدأ بتحميل ببسك السريع.

١ - اكتب البرنامج التالى :

```
File Edit View Search Run Debug Calls F1=Help
Untitled
This program demonstrates the SIN function
The program computes a simulated sine value and compares it
with the standard SIN function. The simulated sine is computed thus:
sin x = x - x^3/3! + x^5/5! - x^7/7! + ...
CLS
PRINT : PRINT "Simulated Sine value and built-in SIN function result"
PRINT : PRINT "x = .1"
x = .1: n = 1: t = x: s = x
```

```
Cont:
n = n + 2
t = (-t * x ^ 2) / (n * (n - 1))
s = s + t
IF ABS(t) <= .00001 THEN GOTO Cont
PRINT "Simulated sine x "; s
PRINT "Built-in SIN function result "; SIN(x)
```

Immediate

Main: <Untitled> Context: Program not running 0001:01

٢ - نفذ البرنامج ولاحظ استخدام دالة SIN في البرنامج. اضغط على اى مفتاح للعودة الى البرنامج.

```
Simulated Sine value and built-in SIN function result
x = .1
Simulated sine x 8.333334E-02
Built-in SIN function result 9.983342E-02
```

Press any key to continue

٣ - من قائمة File اختر New واكتب N لإخلاء الشاشة.

٤ - انتقل الى الدرس المائة والخامس والثلاثين للاستمرار فى تسلسل التعلم.

الدرس المائة والثانى والثلاثون

عبارة SOUND

الوصف

تنتج عبارة SOUND صوتاً بتردد محدد وديمومة محددة من الكمبيوتر، وتكوينها هو كمايلي:

SOUND frequency,duration

التردد عبارة عن تعبير عددي يقع في المدى من 37 الى 32,767، والديمومة هي طول وقت استمرارية الصوت الناتج وهو تعبير عددي يقع في المدى من 0 الى 65,535، ويفسر التردد بأنه عدد من الدورات في الثانية وتفسر الديمومة بأنها دورات الساعة الداخلية للكمبيوتر، وهناك 18.2 دورة للساعة في الثانية الواحدة.

التطبيقات

تستخدم عبارة SOUND في انتاج صوت من الكمبيوتر بطرق عديدة، وفيما يلي بعض الامثلة:

```
SOUND 500, 500
SOUND T%0.2, T%100
SOUND 100,0
```

اخر مثال له ديمومة تساوى صفراً ويتسبب ذلك في ان يتوقف متحدث الكمبيوتر.

عملية تقليدية

هذه العملية توضح استخدام عبارة SOUND، ابدأ بتحميل بيسك السريع،

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls F1=Help
<Untitled>
This program demonstrates the SOUND statement
CLS
PRINT : PRINT
PRINT " This program will produce a sound depending on what key you type."
PRINT " Use the number keys. Type 0 to quit."
CS = ""
DO WHILE (CS <> "0")
  LOCATE 6, 1: INPUT CS
  IF (CS <> "0") THEN
    Snd% = VAL(CS)
    IF Snd% < 9 THEN SOUND Snd% * 120, 25
  END IF
LOOP
'Turn the sound off
SOUND 0, 0
Immediate
Main: <Untitled> Context: Program not running C 00010:011

```

٢ - نفذ البرنامج، لاحظ تأثير عبارة SOUND في البرنامج، اكتب أى رقم واضغط على مفتاح الإدخال (الرقم 9 يقع خارج مدى سماع البشر)، اكتب 0 واضغط على مفتاح الإدخال لايقاف البرنامج.

```

This program will produce a sound depending on what key you type.
Use the number keys, Type 0 to quit.

7 2

```

٣ - من قائمة File اختر New واكتب N لإخلاء الشاشة.

٤ - انتقل الى الدرس المائة وثلاثة للاستمرار فى تسلسل التعلم.

الدرس المائة والثالث والثلاثون

دالة SPACE\$

الوصف

هذه الدالة تعيد سلسلة فراغات لها طول معين. وتكوينها هو كما يلي :

```
SPACE$(numeric expression)
```

يجب ان يقع التعبير العددي في المدى من 0 الى 32,767.

التطبيقات

تستخدم دالة SPACE\$ عندما تكون هناك حاجة الى سلسلة فراغات . ويستخدم البرنامج العينة الموجود في الدرس الثالث دالة SPACE\$ في تحديد وضع جانبي الصندوق على الشاشة. وفيما يلي امثلة اخرى لاستخدام دالة SPACE\$.

```
PRINT 22 SPACE$(22) 22
22
PRINT "'hello" SPACE$(40) "there!"
'hello
```

نصيحة : يمكن ان تستخدم دالتا TAB و SPC في تحديد وضع المخرجات اثناء استخدام عبارة PRINT .

تقليدية

يجرى البرنامج التالي تجاربا مع دالة SPACE\$. ولاختباره ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls F1=Help
Untitled
CLS
One:
INPUT "Enter string to print ", PStr$
IF LEN(PStr$) = 0 THEN GOTO Leave
PStr$ = SPACES(40 - LEN(PStr$)) + PStr$
PRINT PStr$
GOTO One
Leave:
END

Immediate

Main: 'Untitled' Context: Program not running 00010:0003

```

٢ - اضغط على Shift - F5 لتنفيذ البرنامج. اكتب اسطرًا من Mary Had a Little Lamb أو من أي مقطوعة أخرى كرد على "Enter string to print" بعد ثلاثة أو أربعة أسطر اضغط ببساطة على مفتاح الإدخال لإنهاء البرنامج.

```

Enter string to print Mary had a little lamb.
      Mary had a little lamb.
Enter string to print Its fleece was white as snow
      Its fleece was white as snow
Enter string to print And everywhere that Mary went
      And everywhere that Mary went
Enter string to print The lamb was sure to go !
      The lamb was sure to go !
Enter string to print

```

لاحظ كيف يقبل البرنامج مدخلات على هيئة سلسلة مع ملء الأماكن الزائدة الموجودة على اليسار بفراغات وطباعة السلسلة. ويتسبب ذلك في طباعة السلسلة مرحلة ناحية اليمين. ونشجعك على إجراء تعديلات على الشفرة وإجراء تجارب.

٣ - اضغط على أي مفتاح للعودة إلى قائمة البرنامج.

٤ - اضغط على Alt - F ثم اضغط على مفتاح الإدخال واكتب N لازالة البرنامج.

٥ - انتقل إلى الدرس التاسع والأربعين للاستمرار في تسلسل التعلم.

الدرس المائة والرابع والثلاثون

دالة SPC

الوصف

تترك دالة SPC عدداً محدداً من الفراغات في عبارة PRINT او LPRINT . وتكوينها هو كما يلي :

SPC(numeric expression)

ويجب ان يكون التعبير العددي واقعا في المدى من 0 الى 32,767 .

التطبيقات

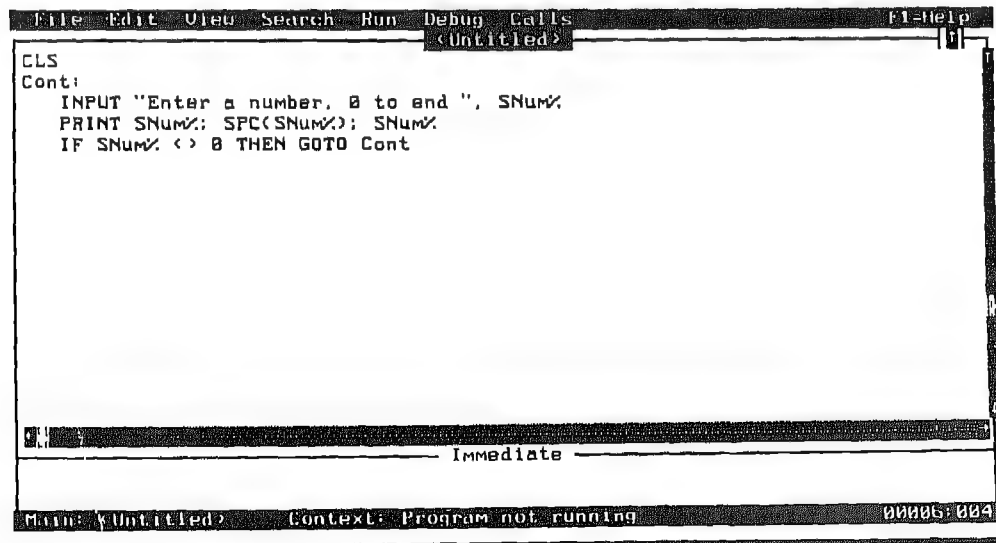
يمكن ان تستخدم دالة SPC مع عبارات PRINT و LPRINT . وفيما يلي بعض الامثلة .

```
PRINT SPC(10) "Here" SPC(10) "Here" SPC(20) "And Here"
PRINT "Minimum value: " SPC(5) Min%
LPRINT "Name" SPC(30) "Address": LPRINT STRING$(41,"-")
```

عملية تقليدية

البرنامج التالي يوضح دالة SPC . ابدأ بتحميل ببسك السريع .

١ - اكتب البرنامج التالي :



٢ - اضغط على Shift - F5 لتنفيذ البرنامج واضغط على ارقام لاختباره.

```

Enter a number, 0 to end 12
12      12
Enter a number, 0 to end 22
22      22
Enter a number, 0 to end 33
33      33
Enter a number, 0 to end 45
45      45
Enter a number, 0 to end 1
1      1
Enter a number, 0 to end 22
22      22
Enter a number, 0 to end 0
0      0

Press any key to continue
    
```

٣ - اضغط على اى مفتاح للعودة الى البرنامج. اضغط على Alt - F ثم اضغط على مفتاح الادخال واكتب N لاخلء الشاشة.

٤ - انتقل الى الدرس السادس للاستمرار فى تسلسل التعلم.

الدرس المائة والخامس والثلاثون

دالة SQR

الوصف

تعيد دالة SQR الجذر التربيعي لعدد معين. وتكونها هو كما يلي :

`SQR(numeric expression)`

التعبير العددي هو عدد اكبر من او يساوى صفراً.

التطبيقات

تستخدم دالة SQR عندما يراد حساب الجذر التربيعي لعدد مثل ما يحدث فى التطبيقات الرياضية والرسومات. وفيما يلي مثال لدالة SQR:

```
FOR T = -3 TO +3 STEP .50
    PRINT SQR(ABS(T))
NEXT T
```

لاحظ ان دالة ABS مستخدمة فى التأكد ان التعبير العددي موجب.

عملية تقليدية

العملية التالية توضح استخدام دالة SQR. ابدأ بتحميل بيسك السريع .

١ - اكتب البرنامج التالى :

```
File Edit View Search Run Debug Calls F1-Help
<Untitled>
' This program solves simple quadratic equations.
CLS
INPUT "Enter a,b,c ": A, B, c
d = B ^ 2 - 4 * A * c
IF d < 0 THEN GOTO PrintMsg

E1 = (-B + SQR(d)) / (2 * A)
EZ = (-B - SQR(d)) / (2 * A)

PrintLine:
PRINT "A = "; A; " B = "; B; " C = "; c
PRINT "Root 1 = "; E1, "Root 2 = "; EZ

END
PrintMsg:
PRINT "Results are complex, and thus ignored."

----- Immediate -----
Main: <Untitled> Context: Program not running
```

٢ - نفذ البرنامج، اكتب 1 على أنه a و 2 على أنه b و -1 على أنه c. لاحظ استخدام دالة SQR في البرنامج. اضغط أى مفتاح للعودة الى البرنامج.

```
Enter a,b,c ? 1,2,-1
A = 1 B = 2 C = -1
Root 1 = .4142135      Root 2 = -2.414214
```

Press any key to continue

٣ - من قائمة File اختر New واكتب N لإخلاء الشاشة.

٤ - انتقل الى الدرس المائة والثامن والأربعين للاستمرار فى تسلسل التعلم.

الدرس المائة والسادس والثلاثون

عبارة STATIC

الوصف

تستخدم عبارة STATIC فى توضيح متغيرات ومنظومات محلية لـ DEF FN او -FUNC TION او SUB لحفظ قيم هذه المتغيرات بين الاستدعاءات ، وتكوينها هو كما يلى :

STATIC variable list

جزء variable list له الشكل التالى :

var1 () AS type , var2 () AS type

تستخدم الاقواس اذا كان اى من var1 او var2 منظومة وتحديد الابعاد اختيارياً . جزء AS type يعطى نوع المتغير ويمكن ان يكون من اى نوع من الانواع البسيطة او التى يحددها المستفيد .

تستخدم عبارة STATIC فى عبارة SUB او FUNCTION او DEF FN فقط . والمتغيرات الموضحة على انها استاتيكية STATIC فى النوال لها أولوية داخل المتغيرات الشاملة بنفس الاسم اذا ما وجد مثل ذلك ، والثوابت الموضحة فى عبارة STATIC تعامل بنفس الطريقة . وعندما توضح عبارة STATIC أياً من SUB او FUNCTION او DEF FN فتعامل كل المتغيرات المحتواة داخله على أنها محلية .

التطبيقات

تستخدم عبارة STATIC اساسا فى التحكم فى الاتصال بالمتغيرات المستخدمة داخل دالة معينة او برنامج فرعى معين . ويمكن ان تستخدم على مستوى توضيح دالة وتسبب فى جعل المتغيرات محلية وتحفظ قيمها بين الاستدعاءات او يمكن ان تستخدم داخل الدالة او البرنامج الفرعى لحفظ قيم متغيرات معينة بين الاستدعاءات ، وفيما يلى بعض الامثلة :

مثال ١

```
DEF FNCountRec(TRec) STATIC
    Count = Count + 1
    ..
END DEF
```

مثال ٢

```
SUB MoveContents(CCrec)
    STATIC MvCnt AS INTEGER, MvErr AS INTEGER
    DIM WriteRec AS CRec
    WriteRec.Fld1 = CCrec.Fld1
    ..
END SUB
```

عملية تقليدية

العملية توضح استخدام عبارة STATIC. ابدأ بتحميل بيك السريع .

١ - اكتب البرنامج التالي :

```
File Edit View Search Run Debug Calls F1=Help
<Untitled>
' This program demonstrates the use of the STATIC statement.
' The program asks for a filename and counts the number of lines
' in that file. The file must be a text file.

DECLARE SUB NoOfLines (FileName$)

CLS
PRINT : INPUT "Enter filename: "; FileName$
IF FileName$ <> "" THEN
    CALL NoOfLines(FileName$)
    PRINT "Number of lines in "; FileName$; " ="; LineCnt%
ELSE
END IF

File Edit View Search Run Debug Calls F1=Help
<Untitled>:NoOfLines
SUB NoOfLines (FileName$) STATIC
    SHARED LineCnt%, line$
    OPEN FileName$ FOR INPUT AS #1
    LineCnt% = 0
    DO WHILE NOT EOF(1)
        LINE INPUT #1, line$
        LineCnt% = LineCnt% + 1
    LOOP
END SUB

Main: <Untitled> Context: Program not running 00000:004
```

٢ - نفذ البرنامج. اكتب اسم ملف نص (LRTRIM.BAS على سبيل المثال) واضغط على مفتاح الادخال ، لاحظ استخدام STATIC في البرنامج.

```
Enter filename: ? LRTRIM.BAS
Number of lines in LRTRIM.BAS = 24
```

Press any key to continue

٣ - اختر New مع اخلاء الشاشة بون ان تحفظ البرنامج.

٤ - انتقل الى الدرس الثالث والأربعين للاستمرار في تسلسل التعلم.

الدرس المائة والسابع والثلاثون

اشباه الاوامر \$STATIC و \$DYNAMIC

الوصف

تستخدم اشباه الاوامر \$STATIC و \$DYNAMIC في توضيح منظومات . والمنظومة الاستاتيكية تكون لها ذاكرة محددة لها اثناء ترجمة البرنامج. اما المنظومة الديناميكية فتكون لها ذاكرة محددة لها اثناء تنفيذ البرنامج فقط. وعندما تستخدم اشباه الاوامر هذه بمهارة فإنها تقدم طريقة قوية للتحكم في كيفية ادارة ذاكرة وقت تنفيذ البرنامج . وهذه الامكانية مهمة عندما يتطلب البرنامج كمية ذاكرة كبيرة وتكون بيئة التطوير او بيئة المقصد لها كمية ذاكرة متاحة اقل من اللازم. والتكوين هو كما يلي :

```
REM $STATIC
REM $DYNAMIC
```

تقدم اشباه الاوامر في عبارة تعليق . يمكن استبدال جزء REM بفاصلة. ولا تكون هناك حاجة دائمة الى توجيهات. كما ان المنظومات تقسم ضمناً الى ديناميكية واستاتيكية اعتماداً على كيفية توضيحها . ارجع الى الدرس الثالث والثلاثين لمناقشة كيفية توضيح منظومة بأنها استاتيكية او ديناميكية.

التطبيقات

تستخدم \$STATIC و \$DYNAMIC في تحسين ادارة ذاكرة وقت التنفيذ . فمع بعض البرامج يعد مكان اكبر للسلاسل عن طريق استبدال المنظومات الاستاتيكية بمنظومات ديناميكية . العبارات الاخرى التي تستخدم مع اشباه الاوامر هذه هي عبارات DIM و REDIM و ERASE . وفيما يلي مثال لذلك :

```
REM $STATIC
CONST X1 = 20, Y1 = 80, X2 = 10, Y2 = 12
DIM WindowTwo(X1,Y1,X2,Y2)
'$DYNAMIC
DIM WindowOne(80,25)
..
REDIM WindowOne(40,25)
..
ERASE WindowOne
```

فى هذا المثال المنظومة WindowTwo هى منظومة استاتيكية. ويجب ان تكون استاتيكية حتى بدون اشباه الاوامر وذلك لاستخدام الثوابت فى الابعاد. اما المنظومة WindowOne فهى ديناميكية ويعاد اعداد ابعادها فيما بعد فى البرنامج. اكثر من هذا فإنها موضحة وهذا يخلو الذاكرة التى تستخدمها المنظومة.

عملية تقليدية

هذه العملية توضح استخدام اشباه الاوامر \$STATIC و \$DYNAMIC فى البرنامج . ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```

File Edit View Search Run Debug Calls Help
<Untitled>
' This program demonstrates the use of $STATIC and $DYNAMIC
' statements. The program declares arrays and deallocates them.
' The memory available before and after is displayed on the screen.

REM $STATIC
DIM W(100, 100)
REM $DYNAMIC
DIM Q(10000)
CLS
PRINT "Amount of free memory ";
PRINT FRE(-1)
REDIM Q(1000)
PRINT "After redimensioning Q ";
PRINT FRE(-1)
ERASE Q
PRINT "After erasing Q ";
PRINT FRE(-1)

Immediate

Name: <Untitled> Context: Program not running 00017:019

```

٢ - نفذ البرنامج. لاحظ استخدام اشباه الاوامر \$STATIC و \$DYNAMIC فى البرنامج.

```

Amount of free memory 262970
After redimensioning Q 298970
After erasing Q 302986

```

Press any key to continue

٣ - ارجع الى البرنامج. احفظ هذا البرنامج كملف نص تحت اسم STA_DYN.BAS مع اخلاء الشاشة.

٤ - انتقل الى الدرس التاسع والثلاثين للاستمرار في تسلسل التعلم.

الدرس المائة والثامن والثلاثون

دالة STICK

الوصف

تستخدم دالة STICK فى قراءة احداثيات اثنين من عصا الالعاب joysticks. وتكوينها هو
كما يلى :

STICK(n)

حيث n هى رقم صحيح يقع بين 0 و 3. والقيم التى تعيدها الدالة لكل قيمة من قيم n هى
كما يلى :

N	القيمة التى تعود
0	احداثى X لعصا الالعاب A.
1	احداثى Y لعصا الالعاب A عند اخر STICK (0) .
2	احداثى X لعصا الالعاب B .
3	احداثى Y لعصا الالعاب B عند اخر STICK (0) .

التطبيقات

دالة STICK تستخدم خصيصا مع عصا الالعاب . ويجب استدعاء STICK (0) قبل
استدعاء اى دالة من دوال STICK (n) وذلك لأن STICK (0) تسجل إحداثيات عصا الالعاب
الاخرى واحداثى X لعصا الالعاب الاولى .

انتقل الى الدرس المائة والحادى والأربعين للاستمرار فى تسلسل التعلم .

الدرس المائة والتاسع والثلاثون

عبارة STOP

الوصف

تسبب عبارة STOP فى انهاء تنفيذ البرنامج. وتكوينها هو كما يلى :

STOP

يمكن استخدام عبارة STOP فى اى مكان فى البرنامج لانهاء البرنامج. وعندما تستخدم عبارة STOP فى بيئة تطوير بيسك السريع فإنها تنهى البرنامج دون ان تغلق اى ملف ولا تعود الى نظام التشغيل. وعندما تستخدم عبارة STOP فى برنامج قائم بذاته فإنها تغلق كل الملفات وتنهى البرنامج وتعود الى نظام التشغيل. وعندما يتم ترجمة البرنامج كبرنامج قائم بذاته بخيار /d او /x او /e فتطبع عبارة STOP رقم السطر الاقرب ما يمكن الى السطر الذى انتهى عنده البرنامج. وعندما لا توجد اى ارقام اسطر فى البرنامج فتطبع عبارة STOP صفرا.

التطبيقات

تستخدم عبارة STOP لانهاء البرنامج طبقا لتمييز المبرمج. وفيما يلى بعض الامثلة :

مثال ١

```
INPUT "Enter file name ";FileName$
..
STOP
..
```

مثال ٢

```
FOR T = 1 TO 20 : PRINT (X*12.2): NEXT T
STOP
```

عملية تقليدية

هذه العملية توضح عبارة STOP . ابدأ بتحميل بيسك السريع .

١ - اكتب البرنامج التالى :

```

File Edit View Search Run Debug Calls FI=Help
<Untitled>
'This program illustrates the use of the STOP statement. The program uses
'the STOP statement instead of the END statement. The result is that the
'program returns to the QuickBASIC development environment with the
'STOP statement highlighted instead of terminating in a normal fashion.
CLS
PRINT "The following is the list of all three character filenames with"
PRINT "extension .BAS."

FILES "777.BAS"
PRINT "Press Enter to continue ": INPUT ts
STOP

Main: <Untitled> Context: <Untitled> 00011:005

```

٢ - نفذ البرنامج. اجب على الملقنات التي تظهر على الشاشة. لاحظ استخدام عبارة STOP في البرنامج لاحظ كذلك كيفية اعادة البرنامج الى بييسك السريع مع زيادة اضاءة عبارة .STOP

```

The following is the list of all three character filenames with
extension .BAS.
C:\QB
BOX .BAS ASC .BAS ABS .BAS
872448 Bytes free
Press Enter to continue
7

File Edit View Search Run Debug Calls FI=Help
<Untitled>
'This program illustrates the use of the STOP statement. The program uses
'the STOP statement instead of the END statement. The result is that the
'program returns to the QuickBASIC development environment with the
'STOP statement highlighted instead of terminating in a normal fashion.
CLS
PRINT "The following is the list of all three character filenames with"
PRINT "extension .BAS."

FILES "777.BAS"
PRINT "Press Enter to continue ": INPUT ts
STOP

Main: <Untitled> Context: <Untitled> 00011:001

```

٣ - انتقل الى البرنامج مع اخلاء الشاشة دون ان تحفظ البرنامج.
٤ - انتقل الى الدرس المائة والسادس والاربعين للاستمرار في تسلسل التعلم.

الدرس المائة والأربعون

دالة STR\$

الوصف

تحول دالة STR\$ تعبيراً عددياً إلى ما يمثله من سلسلة، ومتمم دالة STR\$ هو دالة VAL والتي تحول سلسلة إلى عددها المكافئ لها، وتكوينها هو كما يلي :

`STR$(numeric expression)`

يتحول العدد إلى السلسلة التي تمثله رقماً رقماً، إذا ما احتوى التعبير العددي على نقطة فإنها تتحول كذلك، وإذا كانت قيمة التعبير العددي موجبة فتحتوى السلسلة على فراغات في بدايتها وإلا فإنها تحتوى على إشارة سالبة كقول رمز لها.

التطبيقات

دالة STR\$ هي دالة مريحة عندما تكون هناك حاجة إلى تشكيل طباعة التعبير العددي، ويمكن حذف الفراغات التي توجد في البداية أو النهاية وتصبح المخرجات مضغوطة وسهلة القراءة، وفيما يلي بعض الأمثلة :

```
X% = 2777
PRINT STR$(X%)
```

المخرجات : 2777

```
PRINT STR$(X% / 3)
```

المخرجات : 925.6667

```
PRINT STR$((X% / 3.1) * -1)
```

المخرجات : -895.8065

عملية تقليدية

استخدام دالة STR\$ موضح في هذه العملية، ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls F1=Help
<Untitled>
' This is a demonstration of the STR$ function.
CLS
IVal = 20
TVal = 3.3
UVal = IVal / TVal
SVal$ = STR$(UVal)
PRINT UVal, SVal$
PRINT "Manipulate the result from the STR$ function ..."
PRINT RIGHT$(SVal$, 6), LEFT$(SVal$, 3)

Immediate

Main: <Untitled> Context: Program not running 000009:0440

```

٢ - نفذ البرنامج. لاحظ المخرجات واستخدام دالة STR\$ في البرنامج. تتحول القيمة العددية VVal الى السلسلة (SVal\$) وتنقسم الى جزئين وتطبع .

```

6.060606 6.060606
Manipulate the result from STR$ function ...
060606 6.

Press any key to continue

```

٣ - اضغط على اى مفتاح للعودة الى البرنامج. اضغط على Alt - F واضغط على مفتاح الادخال واكتب N لإخلاء الشاشة.

٤ - انتقل الى الدرس السابع والخمسين للاستمرار فى تسلسل التعلم.

الدرس المائة الحادى والأربعون

دالة STRIG

الوصف

تعيد دالة STRIG حالة اطلاق عصا العاب معينة ، وتكوينها هو كما يلى :

$$STRIG(n)$$

الجزء n هو قيمة عددية صحيحة تقع بين 0 و 7 . ويصف الجدول التالى نتائج كل قيمة من

قيم n.

النتيجة	N
إذا كان الزر السفلى لعصا اللعب مضغوطة منذ آخر (0) STRIG فتكون النتيجة 1 - وإلا فانها تكون 0 .	0
إذا كان الزر السفلى لعصا اللعب لاسفل حالياً عند A تكون النتيجة 1 - وإلا فهي 0	1
إذا كان الزر السفلى مضغوطة عند B منذ آخر (2) STRIG فتكون 1 - وإلا فهي 0.	2
إذا كان الزر السفلى لعصا اللعب على B لاسفل حالياً فتكون 1 - وإلا فهي 0.	3
إذا كان الزر العلوى لعصا اللعب على A مضغوطة منذ آخر (4) STRIG فتكون 1 - وإلا فهي 0 .	4
إذا كان الزر العلوى لعصا اللعب على A مضغوطة حالياً فتكون 1 - وإلا فهي 0.	5
إذا كان الزر العلوى لعصا اللعب على B مضغوطة منذ آخر (6) STRIG فتكون 1 - وإلا فهي 0 .	6
إذا كان الزر العلوى لعصا اللعب على B مضغوطة حالياً فتكون 1 - وإلا فهي 0 .	7

التطبيقات

تستخدم دالة STRIG فى الحصول على معلومات صريحة عن اثنين من عصا الالعاب.
انتقل الى الدرس المائة والثانى والأربعين للاستمرار فى تسلسل التعلم.

الدرس المائة والثاني والأربعون

عبارات STRIG ON و STRIG OFF و STRIG STOP

الوصف

تقوم عبارات STRIG ON و STRIG OFF و STRIG STOP بتمكين والغاء تمكين وإيقاف اصطياد الاحداث على عصا لعب محددة. وتكوين هذه العبارات كما يلي :

```
STRIG(n) ON
STRIG(n) OFF
STRIG(n) STOP
```

قيمة n هي احدى القيم 0 أو 2 أو 4 أو 6 . وتأثير كل قيمة من قيم n موجود فى الجدول التالى:

التأثير	N
اصطياد الزر السفلى على عصا اللعب A.	0
اصطياد الزر السفلى على عصا اللعب B .	2
اصطياد الزر العلوى على عصا اللعب A.	4
اصطياد الزر العلوى على عصا اللعب B.	6

ويمكن عمل اصطياد احداث باستخدام عبارة STRIG (n) ON لزر عصا لعب محددة. واستخدام GOSUB : GOTO STRIG (n) ON يقوم بتشغيل الحدث كما يحدث.

وتلغى STRIG (n) OFF مقدرة اصطياد الاحداث لزر عصا لعب محددة. والاحداث التى تحدث بعد تنفيذ هذه العبارة لا يمكن تذكرتها.

وتوقف STRIG (n) STOP اصطياد الاحداث لزر عصا لعب محددة. والاحداث التى تحدث بعد ذلك يمكن تذكرتها وتشغيلها عندما تنفذ عبارة STRIG (n) ON بعد ذلك لهذا الزر.

التطبيقات

عبارات STRIG ON و STRIG OFF و STRIG STOP تكون اكثر فائدة عند كتابة برنامج يدعم المدخلات باستخدام عصا اللعب. وللمزيد من المعلومات عن اصطيات الاحداث ارجع الى الدرس الثانى والتسعين.

انتقل الى الدرس الحادى والستين للاستمرار فى تسلسل التعلم.

الدرس المائة والثالث والأربعون

دالة STRING\$

الوصف

هذه الدالة تعيد سلسلة لها طول محدد تعد لرمز معين من رموز ASVII. وتكوينها هو كما يلي :

```
STRING$(L,c)
STRING$(L,string expression)
```

حيث L هو تعبير عددي تقع قيمته بين 1 و 32,767 يحدد طول السلسلة المطلوبة. و C هو تعبير عددي تقع قيمته بين 0 و 255 للرمز المراد استخدامه في بناء السلسلة. وتعبر السلسلة هو أى سلسلة ، يستخدم أول رمز من تعبير السلسلة في اعدادها .

التطبيقات

دالة STRING\$ تكون اكثر فائدة في بناء سلاسل عديدة لرموز مختلفة واطوال مختلفة. وأحد مثل هذه الاستخدامات موضح في عينة البرنامج الموجودة في الدرس الثالث حيث استخدمت الدالة في بناء الاسطر العلوية والسفلية للصندوق. وفيما يلي امثلة اخرى لاستخدام دالة \$ STRING .

مثال ١

```
PRINT STRING$(40,".")
.....
```

مثال ٢

```
PRINT STRING$(40,42)
.....
```

مثال ٣

```
PRINT "No! " STRING$(4,46) "Don't do that!"
No! .... Don't do that!
```

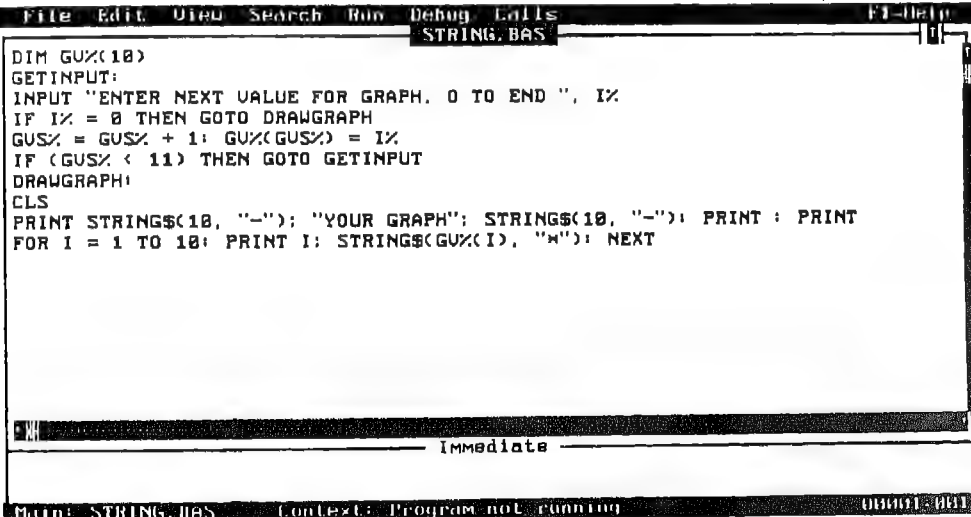
مثال ٤

```
PRINT STRING$(8,"Sure")
SSSSSSSS
```

عملية تقليدية

يقبل البرنامج التالي عشر قيم لأحد الرسومات ويخزنها في منظومة ويرسم الرسم طبقاً لهذه القيم، وتستخدم دالة STRING\$ في رسم اسطر الرسم الأفقية بمجهود بسيط. ابدأ بتحميل ببسك السريع.

١ - اكتب البرنامج التالي :



```
File Edit View Search Run Debug Tools
STRING.BAS
DIM GUX(10)
GETINPUT:
INPUT "ENTER NEXT VALUE FOR GRAPH, 0 TO END ", IX
IF IX = 0 THEN GOTO DRAWGRAPH
GUSX = GUSX + 1: GUX(GUSX) = IX
IF (GUSX < 11) THEN GOTO GETINPUT
DRAWGRAPH:
CLS
PRINT STRING$(10, "-"): "YOUR GRAPH": STRING$(10, "-"): PRINT: PRINT
FOR I = 1 TO 10: PRINT I: STRING$(GUX(I), "x"): NEXT
Main: STRING.BAS Context: Program not running 000001:0001
```

٢ - اضغط على Shift - F5 لتنفيذ البرنامج.

```
ENTER NEXT VALUE FOR GRAPH, 0 TO END 12
ENTER NEXT VALUE FOR GRAPH, 0 TO END 2
ENTER NEXT VALUE FOR GRAPH, 0 TO END 22
ENTER NEXT VALUE FOR GRAPH, 0 TO END 44
ENTER NEXT VALUE FOR GRAPH, 0 TO END 55
ENTER NEXT VALUE FOR GRAPH, 0 TO END 8
```

-----YOUR GRAPH-----

```
1 *****
2 **
3 *****
4 *****
5 *****
6
7
8
9
10
```

Press any key to continue

٣ - اضغط على اى مفتاح للعودة الى قائمة البرنامج.

٤ - من قائمة File اختر Save واكتب STRING.BAS كاسم للملف واحفظ هذا البرنامج.

٥ - انتقل الى الدرس المائة والثلاثين للاستمرار فى تسلسل التعلم.

الدرس المائة والرابع والأربعون

عبارتا SUB و END..SUB

الوصف

تعرف عبارتي SUB و END..SUB برنامج فرعيا . وفيما يلي تكوينها :

```
SUB name (parameter list)
END SUB
```

جزء name هو اسم البرنامج الفرعي المستخدم في استدعاء البرنامج الفرعي. وجزء par-
ameter list اختياري وله التكوين التالي :

```
variable ( ) AS type, variable ( ) AS type
```

تحدد الاقواس اذا ما كان المتغير منظومة (وهذا اختياري) فلا تكون هناك حاجة إلى
الابعاد. ويحدد جزء AS type نوع البيانات التي ينتمى اليها المتغير (وهي اختياري كذلك).
ويمكن للنوع المحدد ان يكون من البيانات البسيطة او من البيانات التي يعرفها المستخدم.

وتمرر المؤشرات بواسطة دليل، واي تغيير يجرى على المؤشرات يؤثر على المتغيرات الاصلية.
وللمزيد من المعلومات عن آلية تمرير المؤشر ارجع الى الدرس الثلاثين.

وتنتهي END SUB البرنامج الفرعي. ويمكن الخروج من البرنامج الفرعي بصورة نهائية
باستخدام عبارة EXIT SUB .

اعتبارات أخرى : فيما يلي قائمة بأشياء أخرى يجب تذكرها عن عبارات SUB.

- على عكس DEF FN او FUNCTION فلا يمكن استخدام SUB في احد التعبيرات.
- يمكن للبرامج الفرعية SUB ان تتسم بسمة الاعداد الذاتية. فيمكنها ان تستدعي نفسها
بنفسها.
- لا يمكن للبرامج الفرعية SUB ان تتداخل.

- لا يمكن للبرامج الفرعية SUB أن تحتوى على عبارات DEF FN او FUNCTION .
- لحفظ قيم المتغيرات المستخدمة فى برنامج فرعى SUB بين الاستدعاءات فتستخدم عبارة .STATIC
- تعتبر كل المتغيرات والمنظومات محلية للبرنامج الفرعى إلا إذا ما عرفت بصورة محددة بأنها مشتركة باستخدام عبارة SHARED.

التطبيقات

تستخدم عبارة SUB بنفس الطريقة مثل DEF FN فى عزل شفرة لها غرض محدد تحديدا جيدا وخالية من الخطأ . وهناك مميزات اضافية لتكوين SUB و END SUB كما سبق توضيحه فى هذا الدرس، وفيما يلى بعض الامثلة.

مثال ١

```
SUB NextRec(CustRec)
    IF EOF(#2) THEN EXIT SUB
END SUB
```

مثال ٢

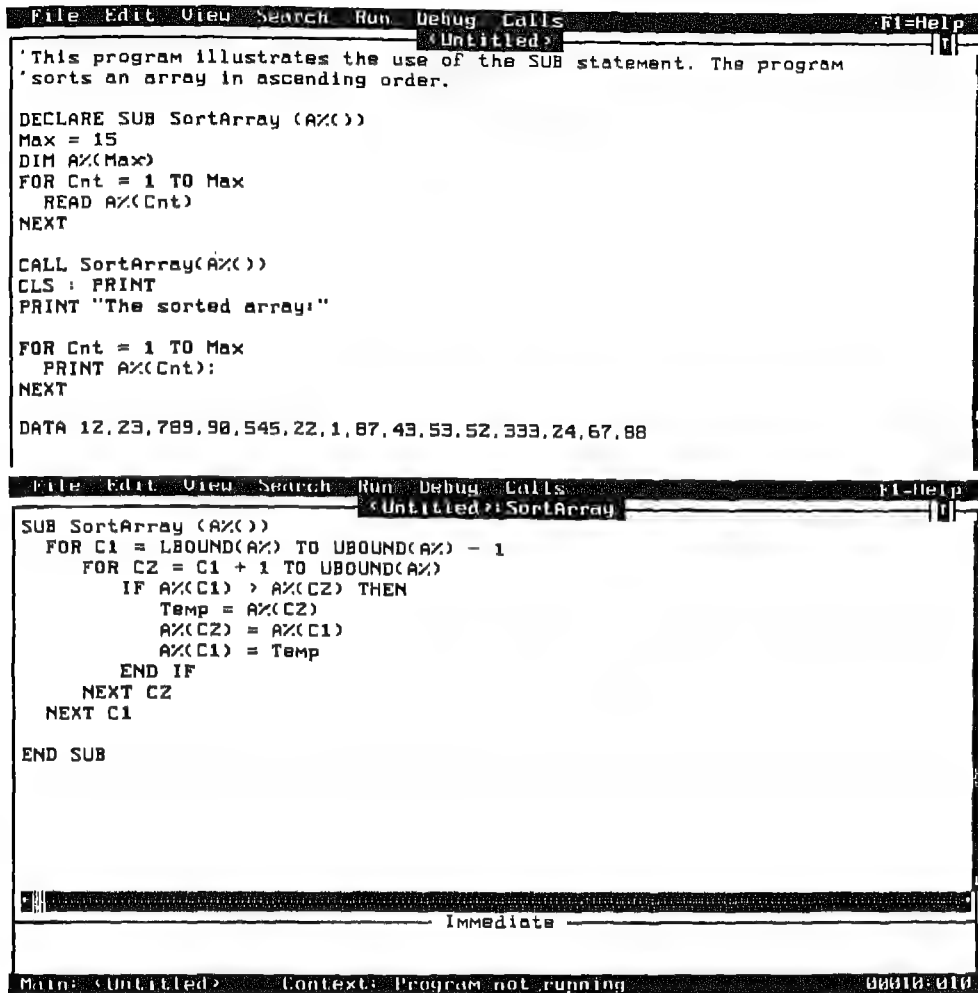
```
SUB WriteRec(CustRec) STATIC
    Count = Count + 1
END SUB
```

يوضح هذا المثال استخدام STATIC فى عبارات SUB . ويتسبب ذلك فى ان المتغير count يحتفظ بقيمته السابقة بين الاستدعاءات .

عملية تقليدية

هذه العملية توضح استخدام عبارة SUB . ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :



```

File Edit View Search Run Debug Calls F1=Help
<Untitled>
' This program illustrates the use of the SUB statement. The program
' sorts an array in ascending order.

DECLARE SUB SortArray (A%( ))
Max = 15
DIM A%(Max)
FOR Cnt = 1 TO Max
    READ A%(Cnt)
NEXT

CALL SortArray(A%( ))
CLS : PRINT
PRINT "The sorted array:"

FOR Cnt = 1 TO Max
    PRINT A%(Cnt):
NEXT

DATA 12,23,789,98,545,22,1,87,43,53,52,333,24,67,88

File Edit View Search Run Debug Calls F1=Help
<Untitled>:SortArray
SUB SortArray (A%( ))
    FOR C1 = LBOUND(A%( )) TO UBOUND(A%( )) - 1
        FOR C2 = C1 + 1 TO UBOUND(A%( ))
            IF A%(C1) > A%(C2) THEN
                Temp = A%(C2)
                A%(C2) = A%(C1)
                A%(C1) = Temp
            END IF
        NEXT C2
    NEXT C1
END SUB

Main: <Untitled> Context: Program not running 00010:010
Immediate

```

٢ - نفذ البرنامج. لاحظ استخدام عبارة SUB في البرنامج.


```
The sorted array:  
1 12 22 23 24 43 52 53 67 87 88 90 333 545 789
```

```
Press any key to continue
```

٣ - احفظ هذا البرنامج كملف نص تحت اسم SUB.BAS مع اخلاء الشاشة.

٤ - انتقل الى الدرس الثانى والخمسين للاستمرار فى تسلسل التعلم.

الدرس المائة والخامس والأربعون

عبارة SWAP

الوصف

تبادل عبارة SWAP محتويات متغيري مؤشرات، وتكوينها هو كما يلي :

```
SWAP var1, var2
```

فإذا كانت $var1 = 10$ و $var2 = 20$ فبعد تنفيذ SWAP تصبح $var1 = 20$ و $var2 = 10$ ، ويمكن ان يكون المتغيران من أى نوع مسموح به فى بيسك السريع، ويجب ان يكون المتغيران من نفس النوع فإذا لم يحدث ذلك فتظهر رسالة خطأ تحدد ان النوع غير متوافق.

التطبيقات

عبارة SWAP هى وسيلة مفيدة عندما يرغب المبرمج فى ابدال قيمتى متغيرين مثال ذلك عند اجراء عملية ترتيب، وفيما يلي بعض الامثلة.

مثال ١

```
SWAP Pct1%, Pct2%
```

مثال ٢

```
PRINT LastName$, FirstName$
$SWAP FirstName$, LastName$
PRINT LastName$, FirstName$
```

مثال ٣

```
TYPE DriverRec
    DriverName AS STRING * 30
    DriverID AS INTEGER
END TYPE
DIM DriverList(20) AS DriverRec, DriverOne AS DriverRec
..
..
SWAP DriverList(10).DriverName, DriverList(1).DriverName
```

عملية تقليدية

العملية التالية توضح استخدام عبارة SWAP. ابدأ بتحميل بيك السريع.

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls File Help
<Untitled>
' This program uses the SWAP statement to sort an array

CLS

Max = 15
DIM A%(Max)
FOR Cnt = 1 TO Max
    READ A%(Cnt)
NEXT

CLS : PRINT
PRINT "The array before sorting:"

FOR Cnt = 1 TO Max
    PRINT A%(Cnt);
NEXT

PRINT : PRINT

PRINT "The sorted array:"
GOSUB SortArray

FOR Cnt = 1 TO Max
    PRINT A%(Cnt);
NEXT
END

SortArray:
FOR C1 = LBOUND(A%) TO UBOUND(A%) - 1
    FOR C2 = C1 + 1 TO UBOUND(A%)
        IF A%(C1) > A%(C2) THEN
            SWAP A%(C2), A%(C1)
        END IF
    NEXT C2
NEXT C1

RETURN

DATA 12,23,789,98,545,22,1,87,43,53,52,333,24,67,88
    
```

Immediate

Main: <Untitled> Context: Program not running 00059:003

٢ - نفذ البرنامج ولاحظ مخرجاته واستخدام عبارة SWAP في البرنامج. اضغط على اى مفتاح للعودة الى البرنامج.

```
The array before sorting:
12 23 789 90 545 22 1 87 43 53 52 333 24 67 88

The sorted array:
1 12 22 23 24 43 52 53 67 87 88 90 333 545 789
```

Press any key to continue

٣ - من قائمة File اختر Save واكتب SWAP.BAS كاسم للملف ثم حدد أن تشكيل الملف نصي واحفظ هذا البرنامج.

٤ - من قائمة File اختر New مع اخلاء الشاشة.

٥ - انتقل الى الدرس المائة والحادي والعشرين للاستمرار في تسلسل التعلم.

الدرس المائة والسادس والأربعون

عبارة SYSTEM

الوصف

تغلق عبارة SYSTEM كل الملفات وتنتهي البرنامج وتعود الى نظام التشغيل. وتكوينها هو كما يلي :

SYSTEM

وعندما تنفذ في برنامج قائم بذاته فيخرج البرنامج الى DOS. وعندما تنفذ من بيئة تطوير بيسك السريع فيتوقف البرنامج. وعندما تنفذ من نافذة فورية Immediate window فيتم الخروج من بيسك السريع.

التطبيقات

تعامل عبارة SYSTEM متماثلة مع عبارة END في بيسك السريع. وتتسبب عبارة SYS-TEM في إغلاق كل الملفات وانهاء البرنامج. واختيار SYSTEM او END يرجع الى المبرمج نفسه. وفيما يلي بعض الامثلة :

مثال ١

SYSTEM

مثال ٢

```
OPEN "Temp.Txt" FOR APPEND AS #10
..
CLS : PRINT "Goodbye .."
SYSTEM
```

عملية تقليدية

هذه العملية توضح استخدام عبارة SYSTEM. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Tools FI-Help
SYSTEM.BAS
' This program illustrates the use of SYSTEM statement. The program uses
' the SYSTEM statement instead of END statement. The result is that the
' the program returns to the QuickBASIC development environment with the
' SYSTEM statement highlighted instead of terminating in a normal fashion.
CLS
PRINT "The following is the list of all files with three character"
PRINT "filenames and extension .BAS"
FILES "???.BAS"
PRINT "Press Enter to continue ": INPUT ts
SYSTEM

Immediate

Main: SYSTEM.BAS Context: Program not running 00001:001

```

٢ - نفذ البرنامج. لاحظ استخدام عبارة SYSTEM في البرنامج. وتبين الشاشة التالية شاشة بيسك السريع بعد انتهاء البرنامج. لاحظ انها لا تكون مختلفة .

```

The following is the list of all files with three character
filenames and extension .BAS
C:\QB
BOX .BAS ASC .BAS ABS .BAS SUB .BAS
STR .BAS A .BAS
2870528 Bytes free
Press Enter to continue
?

```

٣ - ارجع الى البرنامج واختر New ولا تحفظ هذا البرنامج.

٤ - انتقل الى الدرس المائة والثاني والخمسين للاستمرار في تسلسل التعلم.

الدرس المائة والسابع والأربعون

دالة TAB

الوصف

تتسبب دالة TAB في ترحيل مخرجات البرنامج عددا محددا من الفراغات ناحية اليمين. وتستخدم TAB مع عبارات PRINT و LPRINT فقط. وتكوينها هو كما يلي :

TAB(numeric expression)

يجب ان ينتج عن التعبير العددي رقما صحيحا (يقع من المدى 0 الى 32,767) . وينتقل وضع الطباعة الى العمود المحدد في التعبير العددي. فإذا كان العمود ابعد من عرض السطر الحالي فينتقل وضع الطباعة الى السطر التالي .

التطبيقات

دالة TAB مفيدة في التحكم في وضع الطباعة بدقة مرتفعة. ويمكن استخدامها في طباعة تقارير وفي التغذية المرتجعة للبرنامج، وفيما يلي بعض الامثلة :

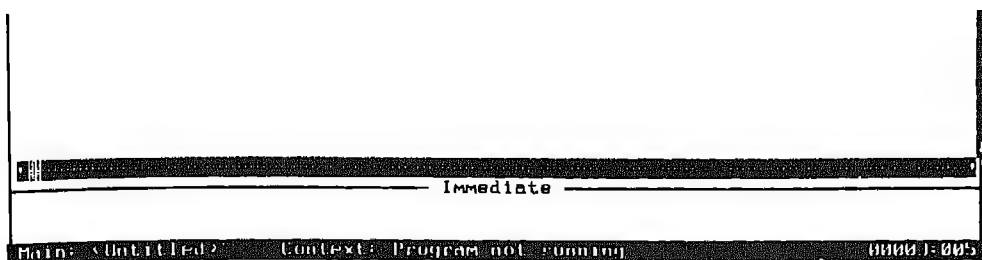
```
PRINT TAB(10) "Name : " TAB(3) LastName$ ", " FirstName$
PRINT TAB(228) "Here you are"
PRINT TAB(1209) "Here you are again !"
LPRINT "File Name:" TAB(10) FileName$
```

عملية تقليدية

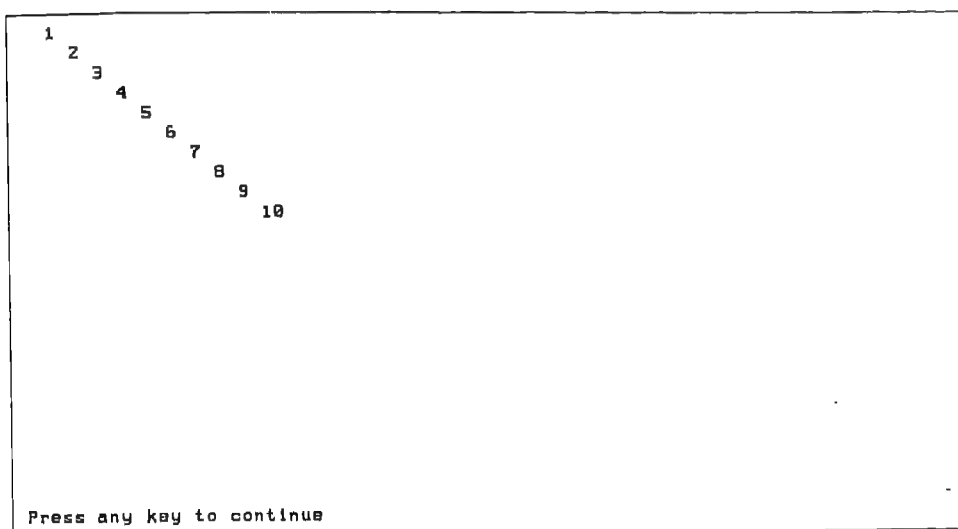
المثال التالي يوضح استخدام دالة TAB. ابدأ بتحميل بيسك السريع .

١ - اكتب البرنامج التالي :

```
File Edit View Search Run Debug Calls F1-Help
<Untitled>
FOR Cnt = 1 TO 10
  PRINT TAB(Cnt * 2); Cnt
NEXT
```



٢ - اضغط على Shift - F5 لتنفيذ البرنامج. لاحظ استخدام دالة TAB لتعبير عددي في تحديد وضع طباعة قيمة عداد FOR .



٣ - اضغط على أى مفتاح للعودة الى قائمة البرنامج. اضغط على Alt - F ثم اضغط على مفتاح الادخال واكتب N لاخلاء الشاشة.

٤ - انتقل الى الدرس المائة والرابع والثلاثين للاستمرار فى تسلسل التعلم.

الدرس المائة والثامن والأربعون

دالة TAN

الوصف

تعطى دالة TAN قيمة ظل الزاوية معبرا عنها بالتقدير الدائرى ، وتكوينها هو كما يلى :

TAN(numeric expression)

تحتسب قيمة TAN بدقة فردية كقيمة تقليدية . وعندما يكون التعبير العددي له دقة مزدوجة فتحتسب قيمة TAN بدقة مزدوجة كذلك. والتعبير العددي هو الزاوية بالتقدير الدائرى . وعندما يحدث سريان زائد لدالة TAN فيتوقف البرنامج إلا إذا كان هناك خطأ فى البرنامج.

التطبيقات

تستخدم دالة TAN عندما يراد حساب ظل الزاوية كما فى حالة الرسومات والتطبيقات الرياضية. وفيما يلى بعض الامثلة.

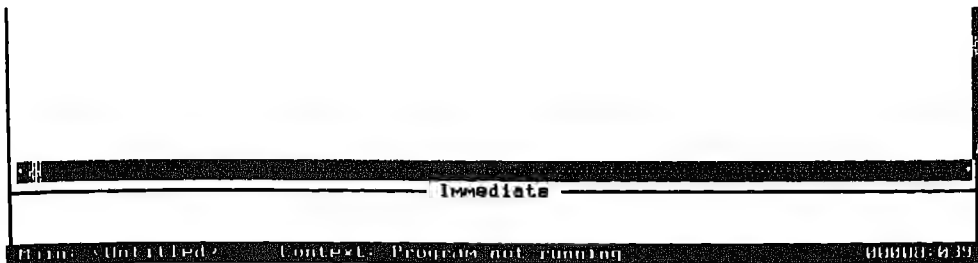
```
Elevation = Base * TAN(Angle)
PRINT "Simulated TAN result "; STan;
PRINT "TAN function result "; TAN(q)
```

عملية تقليدية

العملية التالية توضح استخدام دالة TAN . ابدأ بتحميل ببسك السريع.

١ - اكتب البرنامج التالى :

```
File Edit View Search Run Debug Calls F1=Help
<Untitled>
This program prints the hyperbolic TAN and the result of the
TAN function, tanH x = e^x - e^-x / e^x + e^-x
CLS
INPUT "Enter x "; x
y = EXP(x) - EXP(-x)
z = EXP(x) + EXP(-x)
PRINT "Hyperbolic TAN = "; y / z
PRINT "TAN function result = "; TAN(x)
```



٢ - نفذ البرنامج. اكتب 05 على أنها x (تذكر ان هذه الدالة تكون بالتقدير الدائري) ولاحظ استخدام دالة TAN في البرنامج. اضغط على اى مفتاح للعودة الى البرنامج.

```
Enter x 7.05
Hyperbolic TAN = 4.995835E-02
TAN function result = 5.004171E-02
```

Press any key to continue

٣ - من قائمة File اختر New لإخلاء الشاشة.

٤ - انتقل الى الدرس المائة والثامن والعشرين للاستمرار فى تسلسل التعلم.

الدرس المائة والتاسع والأربعون

دالة وعبارة TIME\$

الوصف

يمكن ان تستخدم الكلمة الرئيسية TIME\$ كدالة او عبارة. وعبارة تضبط TIME\$ ساعة الكمبيوتر لوقت محدد بالنسبة لليوم. وكدالة تعود TIME\$ بالوقت الحالى طبقا لساعة الكمبيوتر. وتكوين كل منهما هو على النحو التالى :

تكوين عبارة TIME\$:

TIME\$=String expression

الجزء	الوصف
TIME\$ string expression	كلمة من كلمات بيسك السريع المحجوزة. الساعة والدقيقة والثانية مفصولة عن بعضها البعض وموضوعة بين علامتى تنصيص مزدوجتين. ويعبر عن الوقت على اساس ساعة بها 24 ساعة بدلا من ان تكون بها 12 ساعة

تكوين دالة TIME\$:

TIME\$

يعيد هذا التكوين الوقت الحالى طبقا لساعة الكمبيوتر الداخلية. والوقت الذى تعيده مبنى على ساعة بها 24 ساعة فى شكل hh:mm:ss حيث hh هى الساعات و mm هى الدقائق و ss هى الثوانى .

التطبيقات

تستخدم عبارة او دالة TIME\$ عندما يحتاج البرنامج ان يتصل بساعة الكمبيوتر. وفيما يلى امثلة لنوعى الاستخدام.

تحديد الوقت باستخدام عبارة TIME\$:

```
TIME$ = "08:30:00"      ' set the clock to 8:30 A.M.
TIME$ = "20:30:00"      ' set the clock to 8:30 P.M.
TIME$ = "07"            ' set the clock to 7:00 A.M.
TIME$ = "10:27"         ' 10:27 A.M.
TIME$ = "00:00:00"      ' initialize to zero
```

الحصول على الوقت باستخدام دالة TIME\$:

```
T$ = TIME$              ' Get the time into T$
PRINT TIME$             ' print the time
T$ = TIME$
Hour = Val(T$)
IF Hour >= 12 THEN D$ = "PM" ELSE D$ = "AM"
PRINT T$ " " D$         ' Print the time and if it is AM or PM
```

عملية تقليدية

العملية التالية تستخدم دالة TIME\$ في توضيح بسيط، ابدأ بتحميل ببسك السريع.

١ - اكتب البرنامج التالي :

```
' This program demonstrates the TIME$ command
T$ = TIME$
Hr% = VAL(TIME$)
M$ = MID$(T$, 4, 2)
Min% = VAL(M$)
IF (Hr% >= 12) AND (Hr% <> 24) THEN
    ap$ = "in the . . PM!"
    IF Hr% <> 12 THEN Hr% = Hr% - 12
ELSE
    ap$ = "in the . . AM !"
    IF Hr% = 24 THEN Hr% = 12
END IF
CLS
SOUND 1100, 8
PRINT "At the sound of the chime the time is "
PRINT Min%: "minutes past ": Hr%: "O'Clock ": ap$
```

Immediate

Main: <Untitled> Context: Program not running 000016:0050

٢ - نفذ البرنامج ولاحظ المخرجات واستخدام TIME\$ فى البرنامج .

```
At the sound of the chime the time is  
54 minutes past 12 O'Clock in the . . PM !
```

Press any key to continue

٣ - اضغط على اى مفتاح للعودة الى البرنامج . من قائمة File اختر Save واكتب TIME.BAS كاسم للملف وحدد أن شكل الملف نصى واحفظ الملف .

٤ - من قائمة File اختر New .

٥ - انتقل الى الدرس الثامن والعشرين للاستمرار فى تسلسل التعلم.

الدرس المائة والخمسون

دالة TIMER

الوصف

تعطى دالة TIMER الوقت بعدد الثوانى المنقضية من الساعة 12.00 am وتكوينها هو كما يلى :

TIMER

تعيد الدالة عددا يتراوح بين 1 و 86,000 (86,000 هو الوقت بالثوانى لعدد 24 ساعة).

التطبيقات

عادة ما تستخدم دالة TIMER مع عبارة RANDOMIZE . وهناك استخدامات عديدة أخرى تتحدد طبقا للتطبيق فقط. وفيما يلى امثلة لاستخدام دالة TIMER .

مثال ١

RANDOMIZE TIMER

مثال ٢

```
Start = TIMER
FOR R = 1 TO 1000
..
NEXT
PRINT (Start-(TIMER)); "Seconds"
```

يستخدم هذا المثال دالة TIMER فى ايجاد طول الفترة التى يستغرقها تنفيذ دورة FOR .. NEXT

عملية تقليدية

هذه العملية تستخدم دالة TIMER فى وضع قيمة ابتدائية لمنتج ارقام عشوائية . وقد تم انتاج البرنامج فى هذه العملية فى الدرس المائة والثالث عشر. ابدأ بتحميل بيسك السريع.

١ - اختر Open وحمل البرنامج الذى سبق انتاجه فى الدرس المائة والثالث عشر،

RANDOM.BAS

٢ - نفذ البرنامج. لاحظ استخدام دالة TIMER لوضع الاساس لمنتج الارقام العشوائية . وهذه طريقة جيدة لوضع اساس منتج الارقام العشوائى لان نتيجة TIMER تكون مختلفة دائما وهذا لضمان ان الاساس يكون فريدا فى كل مرة.

٣ - ارجع الى البرنامج مع اخلاء الشاشة نون ان تحفظ هذا البرنامج.

٤ - انتقل الى الدرس المائة والحادى والخمسين للاستمرار فى تسلسل التعلم.

الدرس المائة والحادى والخمسون

عبارات TIMER ON و TIMER OFF و TIMER STOP

الوصف

عبارة TIMER ON : عبارة تمكن من اصطياد الاحداث لدالة TIMER. والاحداث التى تحدث بعد تنفيذ هذه العبارة يمكن اصطيادها فى الحدث الذى يحدث طبقا لها .

عبارة TIMER OFF : عبارة تطفى من إمكانية اصطياد الاحداث . والاحداث التى تحدث بعد تنفيذ هذه العبارة تهمل ولا يمكن تذكرتها لاستخدامها عندما تنفذ عبارة TIMER ON بعد ذلك.

عبارة TIMER STOP : عبارة توقف اصطياد الاحداث. والاحداث التى تحدث بعد تنفيذ هذه العبارة يتم تذكرتها واصطيادها بعد تنفيذ عبارة TIMER ON تالية.

التطبيقات

تستخدم عبارات TIMER ON و TIMER OFF و TIMER STOP فى اغراض TIMER لاصطياد الاحداث . وتعتمد الاستخدامات على التطبيق نفسه. وفيما يلى بعض امثلة لعبارات TIMER ON و TIMER OFF و TIMER STOP .

```
TIMER ON
ON TIMER(120) GOSUB Process1
..
ON TIMER(1200) GOSUB Process2
..
Process1:
..
RETURN
Process2:
..
TIMER OFF
RETURN
```

يبين المثال السابق استخدام TIMER OFF فى انهاء اصطياد احداث TIMER للبرنامج.

عملية تقليدية

هذه العملية تستخدم البرنامج الذى سبق اعداده فى الدرس الثانى والتسعين. ويستخدم البرنامج عبارة TIMER ON فى تتبع الوقت المنقضى بين ظهور ملقن المدخلات والاستجابة له. ابدأ بتحميل ببسك السريع.

١ - اختر فتح OPEN وحمل البرنامج الذى سبق اعداده فى الدرس الثانى والتسعين ONEVENT.BAS. عدل البرنامج كما هو مبين فى قائمة البرنامج التالية :

```

File Edit View Search Run Debug Calls F1-Help
ONEVENT.BAS
' This program demonstrates the use of the ON KEY(n) GOSUB statement
' and the TIMER ON statement.
' Turn the trapping on for function key F1. Trap the event and
' branch to GetOut. Branch from GetOut to Finit, and end the program.
TIMER ON
KEY(1) ON
ON KEY(1) GOSUB GetOut
ON TIMER(2) GOSUB RingBell
CLS

DO
    InCh$ = INKEY$
LOOP UNTIL True

Finit:
END

GetOut:
PRINT "You did it! You finally found out how to stop this program."
BEEP: BEEP
RETURN Finit

RingBell:
BEEP: BEEP
PRINT "Too late Charlie. You missed your cue."
END

----- Immediate -----
Main: ONEVENT.BAS Context: Program not running 00016:000

```

٢ - نفذ البرنامج. لاحظ انك اذا ما انتظرت لمدة ثانيتين قبل الضغط على F1 فيقوم البرنامج بتصيد هذا الحدث وينفذ GOSUB مع علامة الملاحظة . وفيما يلى عينة للتنفيذ والمخرجات.

Too late Charlie. You missed your cue.

Press any key to continue

٣ - ارجع الى البرنامج مع اخلاء الشاشة دون ان تحفظ هذا البرنامج.

٤ - انتقل الى الدرس العشرين للاستمرار فى تسلسل التعلم.

الدرس المائة والثانى والخمسون

عبارتا TROFF و TRON

الوصف

عبارتا TRON و TROFF تمكن من وتلغى المقدرة على تتبع تنفيذ البرنامج، وتكوينهما هو كما يلى :

```
TRON
TROFF
```

تمكن عبارة TRON من تتبع تنفيذ البرنامج، وتلغى عبارة TROFF المقدرة على تتبع تنفيذ البرنامج، واختيار Trace On من قائمة Debug فى بيسك السريع هو نفسه مثل ادخال عبارة TRON فى البرنامج، وتشهد اضاءة كل عبارة مع تنفيذها، وعندما تنفذ فى برنامج قائم بذاته فتعرض ارقام الاسطر فقط.

التطبيقات

تكرر عبارات TRON و TROFF فى بيئة تطوير بيسك السريع (الاصيغة 4.0) ويمكن تحقيق نفس التأثير من خلال قائمة Dabug، وفيما يلى مثال لذلك :

```
TRON
PRINT: PRINT "Trace on / Trace off Demonstration"
TROFF
```

عملية تقليدية

هذه العملية توضح استخدام عبارات TRON و TROFF، ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالى :

```

File Edit View Search Run Debug Calls F1-Help
<Untitled>
This program demonstrates the use of TRON and TROFF statements.
TRON
TYPE Customer
  CName AS STRING * 40
  Address AS STRING * 40
  LastInvoice AS STRING * 10
  Outstanding AS DOUBLE
END TYPE

DIM CreditPeriod AS INTEGER, Interest AS SINGLE
DIM NextCust AS Customer
CLS
PRINT "Credit Period needs "; LEN(CreditPeriod); " bytes."
PRINT "Interest needs "; LEN(Interest); " bytes."
TROFF
PRINT "Customer record needs "; LEN(NextCust); " bytes."
PRINT "Press Enter to continue."; INPUT t$

```

Immediate

Main: <Untitled> Context: Program not running 00012:019

٢ - نفذ البرنامج. اضغط على مفتاح الادخال لانهاء تنفيذ البرنامج. لاحظ استخدام عبارات TRON و TROFF في البرنامج. لاحظ كذلك كيف تشتت اضاءة عبارة PRINT مع تنفيذ البرنامج.

```

Credit Period needs 2 bytes.
Interest needs 4 bytes.
Customer record needs 98 bytes.
Press Enter to continue.
?

```

```

File Edit View Search Run Debug Calls F1-Help
<Untitled>
This program demonstrates the use of TRON and TROFF statements.
TRON
TYPE Customer
  CName AS STRING * 40
  Address AS STRING * 40
  LastInvoice AS STRING * 10
  Outstanding AS DOUBLE
END TYPE

DIM CreditPeriod AS INTEGER, Interest AS SINGLE
DIM NextCust AS Customer
CLS
PRINT "Credit Period needs "; LEN(CreditPeriod); " bytes."
PRINT "Interest needs "; LEN(Interest); " bytes."
TROFF
PRINT "Customer record needs "; LEN(NextCust); " bytes."
PRINT "Press Enter to continue."; INPUT t$

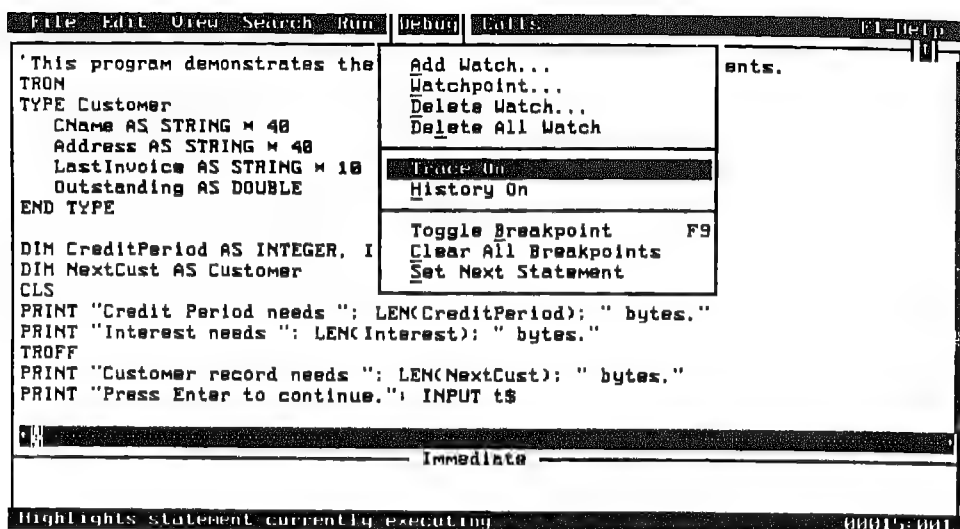
```

Immediate

Main: <Untitled> Context: <Untitled> 00013:059

٣ - اضغط على اى مفتاح للاستمرار .

٤ - الشاشة التالية تبين قائمة Debug التي تقدم طريقة اخرى لتتبع البرنامج. خيار Trace في قائمة Debug تشتت اضاءته. لرؤية ذلك، اختر قائمة Debug من المنقح.



٥ - ارجع الى البرنامج بالضغط على esc واختر New ولا تحفظ هذا البرنامج.

٦ - انتقل الى الدرس الثالث والثلاثين للاستمرار في تسلسل التعلم.

الدرس المائة والثالث والخمسون

عبارتا TYP و END.. TYPE.

الوصف

تستخدم عبارة TYPE فى وصف متغيرات يعرفها المستفيد وتكون على هيئة مجموعة مكونة من عنصر واحد او اكثر. وتكوين العبارة ياخذ الشكل التالى :

```
TYPE
    item1 AS Type
    item2 AS Type
..
END TYPE
```

تسمح عبارات TYPE و END..TYPE بتجميع العناصر. والعناصر تكون مجمعة فى مجموعة وتكون اسماء متغيرات ببسك سريع صحيحة . وجزء type يمكن ان يكون من اى نوع من انواع البيانات الاساسية او من اى نوع من الانواع التى يحددها المستفيد . عناصر السلاسل يكون طولها ثابتاً ويتم وضعها كما يلى :

```
Item1 AS STRING * length
```

جزء length يكون ثابتاً عددياً. (ولا يسمح بالتعبيرات العددية) . ولا يمكن للعناصر ان تكون اسماء منظومات. ويجب تعريف النوع الذى يحدده المستفيد قبل امكانية استخدامه.

التطبيقات

يمكن استخدام عبارة TYPE فى اى مكان فى البرنامج لوصف متغير يعرفه المستفيد. وتكون عبارة TYPE مفيدة لصيغة خاصة فى تعريف متغيرات السجل لاستخدامها مع ملفات وذلك لان البرنامج يمكن كتابته بسهولة اكبر عن كتابته باستخدام عبارة FIELD . وبالنسبة لجعل النوع الذى يعرفه المستفيد متاحا فى البرنامج فيجب ان يوصف النوع الجديد اولاً ثم توضع بعد ذلك المتغيرات بانها من هذا النوع. وفيما يلى بعض امثلة لعبارة TYPE :

```
TYPE NewCust
    CustName AS STRING * 30
    CustCode AS STRING * 12
END TYPE
DIM Customer AS NewCust
```

لا يمكن استخدام النوع الذي يعرفه المستخدم NewCust في البرنامج قبل ان تنفذ عبارة
. DIM

```
READ Customer.CustName, Customer.CustCode
..
TYPE Menu
  MenuItem AS STRING * 35
  Price    AS Single
END TYPE
DIM MenuList(50) AS Menu
```

عملية تقليدية

هذه العملية توضح عبارة TYPE و END.. TYPE في برنامج بسيط . ابدأ بتحميل
بيسك السريع. (ارجع الى البدء في الدرس الثالث وملحق B).

١ - اضغط على Alt - F واكتب N لاختيار New .

٢ - اكتب البرنامج التالي :

```
File Edit View Search Run Debug Calls FI=Help
<Untitled>
'Demonstration of the TYPE..END TYPE statement.
TYPE Acct
  AcctNm AS STRING * 20
  AcctNum AS INTEGER
END TYPE
DIM Account(5) AS Acct
FOR L = 1 TO 3
  READ Account(L).AcctNm, Account(L).AcctNum
NEXT L

CLS
FOR L = 1 TO 3
  PRINT Account(L).AcctNm, Account(L).AcctNum
NEXT L

DATA Mr. Jones, 001, Mr. Burgess, 002, Ms. Stepps, 003

Immediate

Main: <Untitled> Context: Program not running 00016:000
```

٣ - اضغط على Shift - F5 لتنفيذ البرنامج . لاحظ استخدام عبارة TYPE و END TYPE في البرنامج .

Mr. Jones	1
Mr. Burgess	2
Ms. Stepps	3

Press any key to continue

٤ - اضغط على Alt - F ثم اضغط على مفتاح الادخال واكتب N لإخلاء الشاشة .

٥ - انتقل الى الدرس الخامس والسبعين للاستمرار في تسلسل التعلم.

الدرس المائة والرابع والخمسون

دالة VAL

الوصف

تحويل دالة VAL تعبير سلسلة يحتوى على ارقام الى قيمته العددية، وتكوينها هو كما يلي:

VAL(string expression)

تبدأ دالة VAL التحويل من الرمز الموجود على اقصى اليسار فى السلسلة وتستمر فى التحويل حتى يظهر الرمز الذى لا يكون رقما او حتى تنتهى السلسلة ، ويحذف رمز عودة العربة والجداول والفراغات السابقة اثناء عملية التحويل.

التطبيقات

حيث ان تعبيرات السلاسل للارقام لا يمكن استخدامها فى العمليات الحسابية فتستخدم دالة VAL فى استخلاص القيمة العددية من تعبيرات سلاسل وتستخدم هذه القيم فى الحسابات، وفيما يلي بعض الامثلة :

```
PRINT VAL("9.22")
St$ = "111 Louis Ave."
Blk% = VAL(St$)
PRINT "Block " Blk%
```

'Address string
'use VAL function to extract block number
'print block number

المخرجات : Block 111

عملية تقليدية

توضح العملية التالية استخدام دالة VAL، ابدأ بتحميل ببسك السريع

١ - اكتب البرنامج التالى :

```
FILE  EDIT  VIEW  SEARCH  RUN  DEBUG  CALLS  F1-Help
' This program demonstrates the use of the VAL function.

TYPE OneType
  NName AS STRING * 20
  Address AS STRING * 30
  Phone AS STRING * 11
END TYPE

DIM Rec AS OneType

CLS
```

```
DO WHILE UCASE$(Choice$) <> "Y"
  INPUT "Enter name :"; Rec.NName
  INPUT "Address   :"; Rec.Address
  INPUT "Phone     :"; Rec.Phone

  IF VAL(Rec.Phone) = 214 THEN
    PRINT "Sounds like a local area number"
  END IF

  INPUT "Quit ? (Y/N) "; Choice$
LOOP
END
```

----- Immediate -----

File: Untitled Context: Program not running 00000000

٢ - نفذ البرنامج. اكتب Herman Munster واضبط على مفتاح الادخال اكتب -1313 Mock- ingbird Ln. واضبط على مفتاح الادخال . واخيرا اكتب 6789-555-214 واضبط على مفتاح الادخال .

```
Enter name :? Herman Munster
Address    :? 1313 Mockingbird Ln.
Phone      :? 214-555-6789
Sounds like a local area number
Quit ? (Y/N) ? y
```

Press any key to continue

٣ - لاحظ المخرجات واستخدم دالة VAL في البرنامج. اضبط على اى مفتاح للعودة الى البرنامج.

٤ - من قائمة File اختر New لإخلاء الشاشة .

٥ - انتقل الى الدرس المائة والخامس والأربعين للاستمرار في تسلسل التعلم.

الدرس المائة والخامس والخمسون

المتغيرات Variables

الوصف

المتغير variable هو دليل يمكن ان تتغير قيمته اثناء تنفيذ البرنامج، ويمكن ان يكون المتغير فى بيسك السريع لاي نوع من انواع البيانات الاساسية التى سبق تقديمها فى الدرس الثالث والعشرين او لانواع البيانات التى يعرفها المستخدم، ويستخدم اسم المتغير variable name فى الاشارة الى إحدى القيم، وللأسماء الخواص التالية :

- يمكن ان يتكون الاسم من خانة واحدة وحتى 40 خانة كحد أقصى .
- يجب ان يبدأ الاسم باحد الحروف الهجائية (من A الى Z او من a الى z) .
- لا يمكن ان يحتوى الاسم على رموز غير الحروف الهجائية والارقام والنقطة ورموز توضيح النوع : \$ أو % أو # أو & أو ! .
- لا يمكن ان يبدأ الاسم بـ "FN" الا اذا كان استدعاءً لدالة .
- لا يمكن ان يكون الاسم كلمة من كلمات البيسك المحجوزة بالرغم من امكانية ادخال الكلمات المحجوزة داخله، وفيما يلى بعض الامثلة .

```
ProcRead
NewFile
SegWrite
ScreenPrint
```

توضيح المتغيرات : توضيح المتغيرات من النوع الاساسى للبيانات (غير التى يعرفها المستخدم) بإضافة رمز لتوضيح النوع كلاحقة له ، ورموز توضيح النوع المختلفة والانواع التى تحددها للمتغيرات هى ما يلى :

اللاحقة	النوع
\$	سلسلة
%	عددى صحيح

عددي صحيح طويل	&
دقة فردية	!
دقة مزبوجة	#

امثلة :

```
Radius = 10: Radius2% = 13
MaxAmt% = 30000
Prompt$ = "Enter selection (1..4)"
Help$ = "Use the cursor keys to choose and press ENTER to select"
MaxAmt& = 100000
MinVal! = 2.33E-12 MaxVal! = 2.32E+12
Tolerance# = .32D-22: LoadLimit# = 34525576.45555
```

وفيما يلي طرق اخرى لتوضيح المتغيرات :

- عبارة DEF type تتسبب هذه العبارة في معاملة كل المتغيرات التي تبدأ بالحروف الموجودة في عبارة DEF كمتغيرات من هذا النوع الاساس بدون استخدام لاحقة توضيح النوع. وفيما يلي مثال لذلك :

```
DEFINT q,r,s,t
DEFSNG l,m,n
DEFDBL p
```

- عبارة DIM .. AS وفيما يلي مثال لهذه العبارة :

```
DIM Purse AS LONG
DIM Temp AS DOUBLE, Name AS STRING
```

- عبارة COMMON .. AS وفيما يلي مثال لهذه العبارة :

```
COMMON FirstName AS STRING
COMMON Salary AS SINGLE, RatePerHr AS SINGLE
```

- عبارة REDIM .. AS وفيما يلي مثال لهذه العبارة :

```
REDIM Purse AS SINGLE
REDIM Temp AS INTEGER, Tools AS STRING
```

– عبارة AS .. SHARED وفيما يلي مثال لهذه العبارة :

```
SHARED Department AS STRING, Floor AS INTEGER
```

– عبارة AS .. STATIC وفيما يلي مثال لهذه العبارة :

```
STATIC CubicFt AS SINGLE, SalesTax AS DOUBLE
```

ويمكن انتاج انواع بيانات يعدها المستفيد من أنواع بيانات أساسية سبق تعريفها بالطريقة التالية :

– استخدام TYPE و END..TYPE فى تعريف نوع يعرفه المستفيد وعبارة AS .. DIM لتوضيحه .

```
TYPE BinItem
  BinID AS INTEGER
  BinContent AS STRING * 25
  BinLoc AS STRING * 5
  BinStatus AS DOUBLE
END TYPE
DIM InventoryItem AS BinItem
```

– استخدام AS (Size) array DIM فى توضيح منظومة. ويستخدم المثال التالى تعريف الانواع السابقة.

```
DIM InventoryList (100) AS BinItem
```

التطبيقات

المتغيرات هى اكثر الوسائل استخداما فى الاشارة الى بيانات. وطريقة اختيارك لتعريف وتوضيح المتغيرات وانواعها تتحكم فى كيفية معالجة البرنامج للبيانات وتخزينها . ودرجة تعقيد تكوين البيانات المستخدمة فى البرنامج تكون مؤشرا جيدا لدرجة تعقيد البرنامج. وللمتغيرات استخدامات عديدة، وفيما يلي قلة منها :

- مخزن مؤقت للبيانات.
- تخزين بيانات اثناء اجراء الحسابات.

- اختيار المستفيد للقائمة.
- تشغيل عداد للدورات.
- قراءة بيانات من وحدات.
- كتابة بيانات فى وحدات.

عملية تقليدية

توضح هذه العملية استخدام المتغيرات عن طريق استخدام البرنامج الذى سبق تقديمه فى الدرس الثالث (BOX.BAS) يستخدم المتغير Lin% فى هذا البرنامج لحفظ تتبع موقع عنصر البيانات التالى والمراد طباعته داخل الصندوق. ويعمل كعداد لعدد الاسطر. والمتغير WStr\$ هو عنصر بيانات يراد كتابته وبه معلومات يراد كتابتها ، وتعديل WStr\$ واجراء تجارب عليه استمر على النحو التالى :

- ١ - ابدأ ببسك السريع بكتابة QB من عند ملقن DOS واضغط على مفتاح الادخال .
- ٢ - اضغط على Alt - F واكتب 0 لتحميل البرنامج.
- ٣ - اضغط على Tab للذهاب الى الدليل فى جدول حوار البرنامج Load واستخدم مفاتيح الاسهم فى اختيار البرنامج BOX.BAS .
- ٤ - اضغط على مفتاح الادخال لتحميل البرنامج.
- ٥ - انتقل نقطة البداية الى الموقع الموجود فى البرنامج الذى تتحمل فيه السلسلة "Illustrated Quick BASIC" للمتغير WSTR\$ وغير السلسلة بكتابة اسمك بدلا من محتواها .
- ٦ - اضغط على Shift - F5 لتنفيذ البرنامج ، لاحظ طباعة اسمك داخل الصندوق .

تحذير

اختصر اسمك بحيث الا يزيد طوله عن 25 خانة. فلا يطبع البرنامج سلاسل اكبر من ذلك داخل الصندوق. حاول ان تفعل ذلك بنفسك.

- ٧ - اضغط على اى مفتاح للعودة الى قائمة البرنامج.

٨ - اضغط على Alt - F ثم اكتب X للخروج من بيسك السريع. اضغط على b ثم على قضييب
المسافات لاختيار عدم حفظ البرنامج.

٩ - انتقل الى الدرس المائة والثالث والخمسين للاستمرار فى تسلسل التعلم.

الدرس المائة والسادس والخمسون

دالتا VARPTR و VARSEG

الوصف

تستخدم الدوال VARPTR و VARSEG للحصول على الفرع للذاكرة وقطاع متغير محدد. وعندما لا يتواجد المتغير مع استخدام هذه الدوال فيتم انتاج المتغير مع عودة العنوان. وعندما يكون المتغير متغير سلسلة فتعود الدوال بعنوان اول بايت من واصف السلسلة .
دالة VARPTR : تعيد هذه الدالة الفرع للمتغير، وتكوينها هو كما يلي :

```
VARPTR(variable)
```

جزء variable يمكن ان يكون اى متغير ببسك سريع من اى نوع اساسى او يعرفه المستفيد، والفرع الذى يعود يكون داخل القطاع الحالى .

دالة VARSEG : تعيد دالة VARSEG جزء القطاع لعنوان متغير محدد، وتكوينها هو كما يلي :

```
VARSEG(variable)
```

وجزء variable هو نفسه مثل المذكور فى دالة VARPTR، والعنوان الذى يعود هو القطاع المخزن داخل المتغير.

التطبيقات

يتكرر استخدام دالتى VARPTR و VARSEG فى البرمجة مختلطة اللغات. وهناك تطبيقات اخرى تستخدم مع عبارات BLOAD و BSAVE و PEEK و POKE و CALL .
وفيما يلى أمثلة لدالتى VARPTR و VARSEG :

```
DIM Window1(80,25)
DEF SEG = VARSEG(Window1(1,1))
BSAVE "Screen.001", VARPTR(Window1(1,1)), 4078
```


عملية تقليدية

تستخدم العملية التالية البرنامج الذي سبق اعداده في الدرس التاسع، ويخدم البرنامج كذلك كتوضيح لدوال VARPTR و VARSEG ، ابدأ بتحميل ببسك السريع.

١ - اختر open وحمل البرنامج المسمى BLSAVE.BAS .

```

File Edit View Search Run Debug Calls F1=Help
BLSAVE.BAS
'This is a demonstration of the BLOAD and BSAVE statements.

'$DYNAMIC
DIM w(10, 10)
DEF SEG = VARSEG(w(1, 1))
FOR i = 1 TO 10
  FOR i2 = 1 TO 10
    w(i, i2) = i + i2
  NEXT i2, i

'Note the length provided is the number of elements in array w plus 1
'Multiplied by 4.
BSAVE "W.Dat", VARPTR(w(1, 1)), 404
FOR i = 1 TO 10
  FOR i2 = 1 TO 10
    w(i, i2) = 0
  NEXT i2, i
ERASE w
REDIM w(10, 10)
BLOAD "W.Dat", VARPTR(w(1, 1))
CLS

FOR i = 1 TO 2
  FOR i2 = 1 TO 10
    PRINT w(i, i2)
  NEXT i2, i
END

Immediate

Main: BLSAVE.BAS Context: Program not running 00000:001

```

٢ - نفذ البرنامج، لاحظ استخدام VARPTR و VARSEG في البرنامج، تستخدم عبارة DEF SEG دالة VARSEG للحصول على قطاع المتغير وجعله القطاع الحالي ، وتستخدم عبارة BSAVE دالة VARPTR في الحصول على فرع للمتغير وكتابة صورة الذاكرة من هذا الفرع، وتستخدم عبارة BLOAD دالة VARPTR في اعادة تحميل بيانات في موقع الذاكرة .

٣ - ارجع الى البرنامج مع اخلاء الشاشة دون ان تحفظ هذا البرنامج.

٤ - انتقل الى الدرس المائة والسابع والخمسين للاستمرار فى تسلسل التعلم.

الدرس المائة والسابع والخمسون

دالة VARPTR\$

الوصف

دالة VARPTR\$ تعطى تمثيلا على هيئة سلسلة لفرع المتغير المؤشر. وتكوينها هو كما يلي

```
VARPTR$(Variable)
```

يمكن ان يكون المتغير من اى نوع فاذا لم يكن المتغير مستخدما بالفعل فيتم انتاجه وتعيد الدالة تمثيل السلسلة للفرع. فاذا كان المتغير منظومة فيجب ان تكون المنظومة سلسلة متغير ويجب ان تحدد لها ابعاد قبل استخدام دالة VARPTR\$.

التطبيقات

تستخدم دالة VARPTR\$ اساسا مع عبارات PLAY و DRAW . ومع عبارات DRAW ,PLAY تستخدم VARPTR\$ فى تنفيذ سلاسل جزئية . وفيما يلى مثال لدالة VARPTR\$.

```
Subs$ = "p24 p8 18"
PLAY Subs$
PLAY ">>X" + VARPTR$(Subs$)
```

عملية تقليدية

هذه العملية توضح استخدام دالة VARPTR\$ مع عبارة PLAY فى تنفيذ سلسلة جزئية .
ابداً بتحميل ببسك السريع .

١ - اختر open وحمل البرنامج المسمى PLAY.BAS. عدل اول عبارة لتأخذ الشكل الموجود فى القائمة التالية :

```

File Edit View Search Run Debug Calls
PLAY.BAS
' This program demonstrates the VARPTR$ statement
CLS
KEY OFF
Subs$ = "MBL16T155"
PLAY ">X" + VARPTR$(Subs$)
FOR I = 1 TO 4
  READ M$
  PRINT M$
  PLAY M$
NEXT I
KEY ON
END
DATA "o4e8e8e4e8e8e4e8g8c8d8e2
DATA "f8f8f8f8f8e8e8e26e26e8d8d8e8d4g4
DATA "e8e8e4e8e8e4e8g8c8d8e2
DATA "f8f8f8f8f8e8e8e26e26g8g8f8d8c2

Immediate

Main: PLAY.BAS Context: Program not running 00005:022

```

- ٢ - نفذ البرنامج، لاحظ استخدام دالة VARPTR\$ في البرنامج.
- ٣ - ارجع الى البرنامج مع اخلاء الشاشة دون ان تحفظ هذا البرنامج.
- ٤ - انتقل الى الدرس المائة والسابع والعشرين للاستمرار في تسلسل التعلم.

الدرس المائة والثامن والخمسون

عبارة VIEW

الوصف

عبارة VIEW تعرف حدود الشاشة للرسومات، وتكوينها هو كما يلي :

```
VIEW SCREEN (x1,y1)-(x2,y2),color,border
```

يحدد جزء SCREEN ان الحدود المذكورة للشاشة مطلقة وعندما لا يستخدم هذا الجزء فتكون الاحداثيات نسبية لموقع الشاشة الحالى . ويصف جزء (x2 , y2) - (x1 , y1) حدود مجال الرؤية وتكون الاحداثيات قطرية عكسية بحيث انها تكون مستطيلا . وجزء color هو خاصية اللون المستخدمة فى وصف بوابة الرؤية، ويحدد جزء border ان الحدود يجب ان ترسم حول بوابة الرؤية . ويمكن ان تكون الحدود اى قيمة عددية، وعندما تحذف فلا ترسم حدود حول بوابة الرؤية.

وتعرف دالة VIEW منطقة مستطيلة داخل شاشة رسومات معينة بحيث يمكن ان تحتوى المخرجات داخل هذه المنطقة، وهى تشبه فصل جزء من الشاشة لاستخدامه نون التأثير على بقية الشاشة، وعند استخدام عبارة VIEW بدون قوائم فإنها تعرف محتوى الشاشة على انه مجال الرؤية . ويجب ان تقع الاحداثيات المقدمة داخل حدود الثبات المحددة للشاشة.

التطبيقات

تستخدم عبارة VIEW فى تعريف مناطق اصغر داخل شاشات رسومات معينة لمخرجات الرسومات، وعندما تشير مخرجات الرسومات الى احداثيات خارج بوابة الرؤية الحالية فتقطع المخرجات لتناسب البوابة، وتعيد العبارات RUN و SCREEN الشاشة الى حجمها الكامل وتلغى مقدرة عبارات VIEW السابقة . وفيما يلى امثلة لعبارة VIEW :

```
SCREEN 1
VIEW SCREEN (0,0)-(100,100)...1
LINE (10,10)-(50,50)

VIEW (100,10)-(200,200)
FOR C = 1 TO 20
  PSET(C,C)
NEXT
```

عملية تقليدية

هذه العملية توضح استخدام عبارة VIEW في صورة مبسطة، استمر إذا كانت لديك امكانيات رسومات ملونة فقط . ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls
Untitled2
' This program demonstrates the VIEW statement.

SCREEN 1: COLOR 1, 3
CLS
FOR cnt = 1 TO 10
    CIRCLE (65, 65), 5 * cnt, , , , 2
NEXT
VIEW SCREEN (100, 10)-(200, 100), , 1
FOR cnt = 1 TO 10
    CIRCLE (200, 100), 5 * cnt
NEXT

Immediate

Name: Untitled2 Context: Program not running 00014:005

```

٢ - نفذ البرنامج. لاحظ استخدام عبارة VIEW في البرنامج.

٣ - ارجع الى البرنامج مع اخلاء الشاشة لكون ان تحفظ هذا البرنامج.

٤ - انتقل الى الدرس المائة والتاسع والخمسين للاستمرار في تسلسل التعلم.

الدرس المائة والتاسع والخمسون

عبارة VIEW PRINT

الوصف

عبارة VIEW PRINT تعرف حدود الشاشة في الشاشة التالية: وتكوينها هو كما يلي :

```
VIEW PRINT line1 TO line2
```

الجزئين line1 و line2 يعرفان أعلى سطر وادنى سطر في بوابة الرؤية. وعندما يحذفان فتعرف الشاشة على انها بوابة الرؤية.

التطبيقات

عبارة VIEW PRINT مفيدة عندما تريد ان تعرض نصا في جزء من الشاشة دون ان تؤثر على بقية الشاشة، والنص المعروض يدور مستقلا كذلك عن بقية الشاشة بحيث ان النصوص الموجودة في الاجزاء الاخرى من الشاشة لا تتأثر. ولا يمكنك ان تعرف حد العمود لبوابة الرؤية. وفيما يلي امثلة لعبارة VIEW PRINT .

```
CLS
PRINT "Look at the bottom of the screen ...."
VIEW PRINT 10 TO 22
FOR Cnt = 1 TO 20
    PRINT SPC(Cnt); "Scrolling away...."
NEXT
VIEW PRINT 1 TO 5
LOCATE 3,3
PRINT "The LOCATE statement works inside the view port"
```

عملية تقليدية

هذه العملية عبارة عن توضيح بسيط لعبارة VIEW PRENT. ابدأ بتحميل بيسك السريع .

١ - اكتب البرنامج التالى :

```

File Edit View Search Run Debug Calls F1-Help
Untitled
This program demonstrates the use of the VIEW PRINT statement.
The program requests a filename and lists that file.

CLS
FILES "7777.BAS"
INPUT "Enter file to list: "; FileName$

IF FileName$ <> "" THEN
    OPEN FileName$ FOR INPUT AS #1
    VIEW PRINT 18 TO 24: CLS
    InCh$ = INPUT$(1, #1)
    DO WHILE NOT EOF(1)
        IF (InCh$ <> CHR$(13)) THEN PRINT InCh$:
        InCh$ = INPUT$(1, #1)
    LOOP
    CLOSE #1
ELSE
    PRINT : PRINT "Thank you for participating ..."
END IF

```

Immediate

Main: 'Untitled' Context: Program not running 00036:007

٢ - نفذ البرنامج واكتب LINE.BAS ولاحظ دوران القائمة . لاحظ استخدام عبارة VIEW PRINT في البرنامج.

```

C:\QB
BOX .BAS DOX2 .BAS CASE .BAS ASC .BAS
BOX .BAS PLAY .BAS OPEN .BAS LINE .BAS
DRAW .BAS LPOS .BAS SUB .BAS STR .BAS
A .BAS
2029568 Bytes free
Enter file to list: 7 LINE.BAS

t% = AX(Cnt)
'Compute the height of the bar
yZ = t% * 2
'Draw the box
LINE (x1 + 5, y1)-(x2, yZ), 2, BF
'Move both x-coordinates to the right
x1 = x1 + 20: x2 = x2 + 20
NEXT

DATA 13,25,44,65,32,22,8,5,71,8

Press any key to continue

```

٣ - ارجع الى البرنامج مع اخلاء الشاشة دون ان تحفظ هذا البرنامج.
٤ - انتقل الى الدرس المائة والثاني والستين للاستمرار في تسلسل التعلم.

الدرس المائة والستون

عبارة WHILE.. WEND

الوصف

تستخدم عبارة WHILE.. WEND فى تكرار تنفيذ سلسلة من عبارات البرنامج حتى لايتحقق شرط معين، وتكوينها هو كما يلى :

```
WHILE condition
  program statements
WEND
```

والشرط هو تعبير بوليان يقوم على أنه صحيح أو خطأ. وتنفذ عبارات البرنامج الموجود داخل مجموعة WHILE.. WEND مرات ومرات طالما أن الشرط متحقق. وعندما تصبح قيمة الشرط خطأ فيستمر التنفيذ بالعبارة التى تلى عبارة WEND، ويمكن أن تتداخل عبارات WHILE.. WEND إلى أى مستوى. وعندما لايتوافق عدد من عبارات WHILE مع عدد من عبارات WEND فينتج المترجم رسالة خطأ، فإذا تعدت عبارات WEND عبارات WHILE فتظهر رسالة خطأ بوجود WEND بدون WHILE، وإذا تعدت عبارات WHILE عبارات WEND فتظهر رسالة خطأ تدل على ذلك أيضاً.

ويجب عمل تداخل أى مكون تحكم بحذر لضمان الحفاظ على سلامة منطق البرنامج. واحدى طرق عمل ذلك هى فحص مجموعات عبارات البرنامج الموجودة داخل مكون التحكم المتداخل بعناية.

التطبيقات

عبارة WHILE.. WEND هى وسيلة مفيدة جداً فى تكرار تنفيذ مجموعة أسطر من أسطر البرنامج بون إعادة كتابتها مرات ومرات داخل البرنامج. وفيما يلى أمثلة لذلك :

مثال ١

```
Q% = 0
WHILE (Q% < 10)
  GOSUB GS1
  Q% = Q% + 1
WEND
```

مثال ٢

```
WHILE (Ret$ <> "Q")
  ..
  WHILE (Ret2$ <> "2")
    ..
  WEND
WEND
```

يوضح هذا المثال تداخل عبارة WHILE.. WEND.

عملية تقليدية

هذه العملية توضح استخدام عبارة WHILE.. WEND. ابدأ بتحميل بيسك السريع.

١ - أكتب البرنامج التالي :

```
File Edit View Search Run Debug Calls F1=Help
<Untitled>
' This program demonstrates the WHILE..WEND statement.
' The program computes Fibonacci numbers, numbers that are sums of
' the preceding two numbers.
CLS
PRINT : PRINT "Fibonacci numbers generator"
PRINT : PRINT
INPUT "Enter the first number (Integer)"; N1%
INPUT "Enter the next number (Integer)"; N2%
INPUT "Enter the number of Fibonacci numbers that you want "; Cnt%
PRINT
CZ% = 3
IF Cnt% > 0 THEN
  PRINT 1, N1%
  PRINT 2, N2%
  WHILE CZ% < Cnt% - 2
    Temp% = N1% + N2%
    PRINT CZ%, Temp%
    N1% = N2%; N2% = Temp%
    CZ% = CZ% + 1
  WEND
ELSE
  PRINT "Thank you for participating !"
END IF
```

Immediate

Program not running

٢ - نفذ البرنامج. لاحظ استخدام عبارة WHILE.. WEND في البرنامج. أكتب 1 كأول رقم و 2 كالرقم الثاني و 12 كعدد لأرقام فيبوناكي الذي تريده.

```

Fibonacci numbers generator

Enter the first number (Integer)? 1
Enter the next number (Integer)? 2
Enter the number of Fibonacci numbers that you want ? 12

1          1
2          2
3          3
4          5
5          8
6          13
7          21
8          34
9          55
10         89
11        144
12       233

Press any key to continue
    
```

٣ - اضغط على أى مفتاح للعودة إلى البرنامج، اختر New من قائمة File واكتب N لإخلاء الشاشة.

٤ - انتقل إلى الدرس الرابع والثلاثين للاستمرار فى تسلسل التعلم.

الدرس المائة الحادى والستون

عبارة WIDTH

الوصف

تحدد عبارة WIDTH طول السطر لأحد الملفات أو لإحدى الوحدات، وتكوينها هو كما يلى :

```
WIDTH cols, lines
WIDTH #file num|device, width
WIDTH LPRINT width
```

فى التكوين الأول جزء cols يحدد عدد الأعمدة فى المخرجات ويحدد جزء lines عدد الأسطر فى المخرجات، والوحدة التى تتأثر هى الشاشة والقيم المسموح بها للأعمدة هى 80 و 40، والقيمة التقليدية هى 80. ويمكن أن تفترض لجزء الأسطر أى قيمة من القيم التالية : 25 أو 30 أو 43 أو 50 أو 60.

وفى التكوين الثانى يحدد القضيب الرأسى (|) إما رقم الملف #filenum أو الوحدة التى يمكن استخدامها. ورقم الملف هو الرقم المحدد للوحدة كملف مفتوح بعبارة OPEN. والوحدة هى وحدة DOS صحيحة ويعبر عنها بسلسلة مثل "LPT1"، وعندما يستخدم رقم الملف فيكون أثر WIDTH فورياً وعندما تستخدم الوحدة فيبدأ تأثير WIDTH بعد عبارة OPEN التالية على الوحدة.

وفى التكوين الثالث يتأثر طابع الأسطر ويكون تأثير عبارة WIDTH فورياً، وجزء width هو عدد الأعمدة للطابع.

التطبيقات

عبارة WIDTH تكون مفيدة عندما يراد إجبار المخرجات لتأخذ عرضاً محدداً سواء كان ذلك بسبب قيود الوحدة أو طبقاً لمتطلبات التطبيق، وفيما يلى أمثلة لعبارة WIDTH.

مثال ١

```
OPEN "LPT1:" FOR OUTPUT AS #1
WIDTH #1, 55
```

مثال ٢

WIDTH LPRINT 60

مثال ٣

WIDTH 25, 40

عملية تقليدية

هذه العملية توضح عبارة WIDTH على طابع الأسطر. استمر إذا كان لديك طابع متصل بجهاز الكمبيوتر المتاح لك فقط. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls F1=Help
<Untitled>
' This program demonstrates the use of the LINE INPUT and LINE INPUT #
' statements. The program accepts text, writes it to a file,
' reads the text back and writes it to the screen.

CLS
PRINT "Enter text: (Enter blank line to terminate)"

OPEN "Notes.001" FOR OUTPUT AS #2

Line$ = " "
DO WHILE Line$ <> ""
    LINE INPUT ">"; Line$
    WRITE #2, Line$
LOOP
CLOSE #2

CLS
PRINT "What you typed in is as follows: "
WIDTH LPRINT 55

OPEN "Notes.001" FOR INPUT AS #2
DO WHILE NOT EOF(2)
    LINE INPUT #2, Line$
    LPRINT Line$
LOOP

CLOSE #2

----- Immediate -----
Main: <Untitled> Context: Program not running 00036.009

```

٢ - نفذ البرنامج ولاحظ استخدام عبارة WIDTH والتأثير على مخرجات الطابع.

٣ - اكتب : This example shows the effect of the WIDTH LPRINT command

واضغط على مفتاح الادخال. اضغط على مفتاح الادخال مرة أخرى لإنهاء البرنامج. يظهر السطر الذي كتبته كسطرين مطبوعين.

٤ - ارجع إلى البرنامج مع اخلاء الشاشة دون أن تحفظ هذا البرنامج.

٥ - انتقل إلى الدرس التاسع والتسعين للاستمرار في تسلسل التعلم.

الدرس المائة والثانى والستون

عبارة WINDOW

الوصف

تسمح عبارة WINDOW بإعادة تعريف احدثايات الشاشة بنظام احدثايات منطقى جديد. وتكوينها هو كما يلى :

WINDOW SCREEN (x1,y1)-(x2,y2)

جزء (x1, y1) - (x2, y2) يحدد حدود الاحداثيات الجديدة لمخرجات الرسومات (x1, y1) تمثل الركن العلوى الايسر وتمثل (x2, y2) الركن السفلى الايمن. ويحول جزء SCREEN الاحداثيات بحيث إن القيم تتحرك من صغير إلى كبير ومن القاعدة إلى القمة.

التطبيقات

تستخدم عبارة WINDOW عندما تحتاج مخرجات الرسومات إلى نظام احدثايات مختلف عن الاحداثيات الطبيعية لحالة معينة للشاشة. ويمكن تعريف مدى الرؤية داخل نظام الاحداثيات الجديد باستخدام عبارة VIEW. وعندما تستخدم عبارة WINDOW بدون قوائم فإنها تلغى مقدرة أى إعداد سبق اعداده بواسطة WINDOW وفيما يلى أمثلة لعبارة WINDOW.

مثال ١

```
SCREEN 1
WINDOW (-50,1)-(50,50)
LINE (-50,1)-(-25,25)
..
```

مثال ٢

```
WINDOW SCREEN(-50,1)-(50,50)
LINE (-50,1)-(-25,25)
..
```

يرسم المثال الأول سطرًا من القاعدة إلى القمة ويرسم المثال الثاني سطرًا من القمة إلى القاعدة. وهذا بسبب أن الاحداثيات تذهب طبيعياً من صغير في القمة إلى كبير في القاعدة وفي المثال الثاني يعكس جزء SCREEN ذلك.

عملية تقليدية

عبارة WINDOW موضحة في صورة مبسطة في هذه العملية. استمر إذا كانت لديك إمكانيات رسومات ملونة فقط. ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :

```

File Edit View Search Run Debug Calls F1=Help
<Untitled>
' This program demonstrates the WINDOW statement.
SCREEN 1: COLOR 1, 3
CLS
FOR Cnt = 1 TO 10
    LINE (65, 65)-(100, 10 * Cnt)
NEXT
WINDOW (-100, -150)-(-150, 200)
LINE (-100, -150)-(-150, 200), 1, B
FOR Cnt = 1 TO 10
    LINE (-1, -1)-(-100, 10 * Cnt)
NEXT
Immediate
Main: <Untitled> Context: Program not running 00015:00%
```

٢ - نفذ البرنامج، لاحظ استخدام عبارة WINDOW في البرنامج.

٣ - ارجع إلى البرنامج واحفظ هذا البرنامج كملف نصي تحت اسم ملف WINDOW.BAS.

٤ - انتقل إلى الدرس المائة وخمسة للاستمرار في تسلسل التعلم.

الدرس المائة والثالث والستون

عبارتا WRITE و WRITE

الوصف

عبارة WRITE : تكتب العبارة البيانات على الشاشة. وتكوينها هو كما يلي :

WRITE expression list

جزء expression list هو قائمة بعناصر البيانات المراد كتابتها على الشاشة. وتستخدم فواصل كمحددات في هذا الجزء وتوضع سلسلة البيانات بين علامتي تنصيص مزدوجتين. تكتب البيانات العددية على الشاشة بدون فراغات سابقة لها أو تابعة لها. وتكتب عبارة WRITE تسلسل عودة العربة وتغذية السطر بعد expression list.

عبارة WRITE# : تكتب العبارة البيانات في ملف. وتكوينها هو كما يلي :

WRITE #filename, expression list

جزء filename هو رقم الملف المحدد للملف في عبارة OPEN. وجزء expression list هو قائمة بعناصر البيانات المراد كتابتها في الملف. وتوفر سلسلة البيانات داخل علامتي تنصيص مزدوجتين وتستخدم فواصل داخل قائمة التعبير كمحددات. وتضيف عبارة WRITE# فواصل كمحددات عند الكتابة في الملف. وتكتب البيانات العددية بدون فراغات سابقة لها أو تابعة لها. ويكتب تسلسل حركة العربة وتغذية السطر بعد expression list. وبدون قائمة التعبير تكتب عبارة WRITE# سطرًا فارغًا في الملف.

التطبيقات

عبارة WRITE : تستخدم العبارة عندما تكون هناك حاجة إلى تحكم أقل في المخرجات وذلك لأنها لا تقدم إمكانيات تشكيل مثل عبارة PRINT. وفيما يلي بعض الأمثلة :

```
T$ = "You don't have any pie neither !"
Remainder = (100/33) - 3
WRITE 12.0,T$,Remainder
```

عبارة WRITE# : عبارة WRITE# مفيدة في انتاج ملفات مع عناصر بيانات محددة.
أحد استخدامات مثل هذا الملف هو استخدام عبارة INPUT# في تحميل القيم داخل المتغيرات مباشرة، وفيما يلي بعض الأمثلة :

```
OPEN "Train.Log" FOR OUTPUT AS #4
WRITE #4 Name$, TrnType$, TrnScore
WRITE #4
```

في هذا المثال تكتب عبارة WRITE# بدون قائمة تعبير سطرأ فارغاً في الملف.

عملية تقليدية

هذه العملية توضح عبارات WRITE و WRITE# . ابدأ بتحميل بيسك السريع.

١ - اكتب البرنامج التالي :

```
File Edit View Search Run Debug Tools Help
<Untitled>

'The following program demonstrates the WRITE and WRITE # statements.
'The program generates data to write to a file and writes the same data
'to the screen using WRITE and to the disk using WRITE #. The disk file
'is viewed using the DOS TYPE command.

CLS
PRINT "Gentlemen, behold the difference between PRINT and WRITE !"
PRINT
var1$ = "No way Jose!"; var2$ = "Dem's de facts."; var3 = 248
PRINT "The PRINT statement:"
PRINT var1$, var2$, var3
PRINT "The WRITE statement:"
WRITE var1$, var2$, var3

OPEN "WP.Txt" FOR OUTPUT AS #1
WRITE #1, var1$, var2$, var3
CLOSE #1

Immediate

Main: <Untitled> Context: Program not running 00017:009
```

٢ - نفذ البرنامج. لاحظ استخدام عبارات WRITE و WRITE# في البرنامج.

Gentlemen, behold the difference between PRINT and WRITE !

The PRINT statement:
No way Jose! Dem's de facts. 248

The WRITE statement:
"No way Jose!", "Dem's de facts.", 248

C:\QB>type wp.txt
"No way Jose!", "Dem's de facts.", 248

C:\QB>

٣ - أخرج من بيסק السريع دون أن تحفظ البرنامج. وعند ملقن DOS اكتب WP.TXT واضفط على مفتاح الادخال. تبين الشاشة (بملقن مختلف) ما يلي :

C:\BASIC\QB.400\ILL.QB>type wp.txt
"No way Jose!", "Dem's de facts.", 248
C:\BASIC\QB.400\ILL.QB>

٤ - انتقل إلى الدرس الثالث والخمسين للاستمرار في تسلسل التعلم.

ملحق A

الاصطلاحات وتعريفاتها

الوصف

فى هذا الملحق تعرف بعض مصطلحات الكمبيوتر والبرمجة المستخدمة فى هذا الكتاب. وكلمات بيسك السريع المحجوزة والتي كانت موضوع محتويات الأجزاء معرفة فى أول مقطع لكل جزء مناظر لها. وعلى هذا لم يتكرر تعريف هذه الكلمات فى هذا الملحق.

الاصطلاح	التعريف
ASCII	اختصار الشفرة النمطية الأمريكية لتبادل المعلومات American Standard Code of Information Interchange. وتستخدم كنمطية لتشكيل الرموز والتنقيط وشفرات تحكم البيانات المستخدمة بواسطة الكمبيوتر.
Boolean expression	تعبير بولياني وهو تعبير تكون نتيجته صحيح TRUE أو خطأ FALSE.
Buffer	ذاكرة احتياطية وهى آلية تسمح لشيئين أن يعملوا معاً عندما يكون أحدهما أقل من الآخر.
Clipboard	لوحة القص، وفى بيسك السريع تكون لوحة القص هى المكان الذى يخزن فيه النص بعد عملية قص cut وقبل عملية لصق paste.
Control structures	تكوينات تحكم وهى آليات مقدمة فى لغات البرمجة تسمح للمبرمج بالتحكم فى منطق البرنامج وسريان البيانات، وعبارات بيسك السريع التالية : IF.. THEN.. ELSE و FOR.. NEXT و DO LOOP و WHILE.. WEND هى التى تعمل كهذه الآليات.
Data file	ملف بيانات، وهو ملف معلومات مرتب بطريقة يسبق تحديدها.
Debugging	عملية تصحيح أخطاء البرنامج.
Defaults	هذه هى قيم تستخدم عندما لا يريد المستفيد تحديد قيمة جديدة، أى إنها قيم تقليدية.

الاصطلاح	التعريف
Dialog Boxes	صناديق حوار وهى طريقة تداخل المستخدم مع نظم البرامج فى بيسك السريع. ويمكن أن يحتوى صندوق الحوار على خيارات للاختيار منها ومعلومات تقدم لنظم البرامج وتنفيذ اجراءات بواسطة نظم البرامج وتأكيد العملية وما إلى ذلك.
Directory	دليل وهو قائمة بالملفات المخزنة على القرص.
Extension	توسع يكون مكوناً من رمز واحد أو اثنين أو ثلاثة ويستخدم كلاحقة لاسم الملف. وعادة ما يستخدم فى تحديد نوع الملف.
Filename	اسم ملف وهو اسم يعطى بواسطة المستخدم لملف بيانات. ولا يزيد طول اسم الملف عن ثمانية رموز مع اضافة ثلاثة رموز كتوسع لاسم الملف (لاحقة). وتفصل اللاحقة عن اسم الملف بواسطة نقطة فى الصورة FILENAME.EXT.
Function Keys	مفاتيح وظائف وهى المفاتيح من F1 إلى F10 (أو F12) الموجودة على لوحة المفاتيح. وأحياناً تتحدد لها بعض الوظائف بواسطة نظم البرامج.
Hardware	نظم مكونات وهى المعدات التى يتكون منها الكمبيوتر.
Literals	ثوابت وهى قيم لا تستخدم كتمثيل لقيم أخرى، مثال ذلك 1 أو 2 أو 4444 أو "Newark".
Logged drive or directory	وحدة مفتوحة أو دليل مفتوح وهى الوحدة الحالية أو دليل القرص الحالى.
Memory	ذاكرة وهى منطقة تخزين مؤقتة للكمبيوتر. وتستخدم معظم أجهزة الكمبيوتر ذاكرة من أشباه الموصلات ويشار إليها بأنها ذاكرة اتصال عشوائى أو RAM. ويفقد الكمبيوتر المعلومات المخزنة فى الذاكرة عندما يقطع عنه التيار. وتحمل برامج الكمبيوتر وتقيم فى ذاكرة الكمبيوتر عندما يتم تنفيذها.
Module	جزء ويشير فى بيسك السريع إلى مجموعة من أسطر برنامج تخزن فى ملف منفصل.

الاصطلاح	التعريف
Nesting	تداخل وهي طريقة لأحد تكوينات التحكم الذي يحتوى على تكوين تحكم آخر يشبهه أو من نوع آخر.
Operating system	نظام تشغيل وهو برنامج كمبيوتر مصمم للعمل مع الكمبيوتر كسطح بينى بينه وبين نظم برامج التطبيقات.
Pathname	اسم مسار وهو اسم مشغل أو دليل يخزن فيه الملف.
Prompt	ملقن وهو عرض للشاشة يحدد أن النظام معد لإجراء معين من المشغل أو لتقديم معلومات إلى المشغل.
Pull-down Menu	قائمة السحب لأسفل وهي نوع من القوائم يعلق لأسفل من اختيار قائمة أخرى.
RAM	ذاكرة اتصال عشوائى. انظر Memory.
Reseeding	اعادة وضع قيمة للأساس وهي وضع قيمة ابتدائية لمنتج أرقام عشوائية. وهي مثل seeding.
Root Directory	دليل الجذر وهو الدليل الأسمى أو أعلى دليل موجود على القرص.
Run-time Error	خطأ وقت التنفيذ وهو خطأ فى البرنامج يحدث أثناء تنفيذ البرنامج.
Scrolling	دوران وهو حركة أفقية أو رأسية لأحد النصوص على الشاشة.
Software	نظم برامج وهي البرامج (تعليمات الكمبيوتر) وملفات البيانات المقيمة على وسط ممغنط أو على أوراق أو فى نظم مكونات الكمبيوتر.
Stack space	مكان الرصة وهي منطقة فى الذاكرة تخزن فيها تعليمات البرنامج.
Statement	مجموعة عبارات وهي مجموعة من أسطر البرنامج محصورة داخل أحد تكوينات التحكم.
Block	دليل جزئى وهو دليل أو مسار ملف ينتمى إلى دليل مرتفع المستوى.
Subdirectory	ثابت رمزى وهو اسم متغير فى بيسك السريع يشير إلى قيمة لاتتغير أثناء تنفيذ البرنامج.
Symbolic Constant	وظائف يمكن أن تحول من الوضع on إلى الوضع off والعكس.
Toggles	تتبع وهي عملية يمكنك أن ترى بواسطتها ما يحدث عند كل خطوة أثناء تنفيذ البرنامج.
Tracing	

الاصطلاح	التعريف
Watch Points	نقاط ملاحظة وهي مواقع محددة في البرنامج يتم ملاحظتها لتصحيح الأخطاء أو لتحسين الأداء.
Wildcard	رمز خاص يمثل أى قيمة قانونية، يستخدم رمز * فى تمثيل أى اسم ملف قانونى فى DOS.

ملحق B

استخدام منقح بيسك السريع

الجزء الرئيسى لبيئة تطوير بيسك السريع هو المنقح القوى والذي له شاشة ذكاء كاملة، ويقدم هذا الملحق عرضاً عاماً للمنقح، ويعودك هذا الملحق على سمات منقح بيسك السريع التالية :

- كتابة برنامج.
- تنفيذ برنامج.
- حفظ برنامج.
- استخدام صناديق الحوار.
- تحميل برنامج.
- كتابة برامج فرعية.
- الانتقال بين البرامج الفرعية.
- القص واللصق.
- الایجاد والاستبدال.
- التتبع وتحديد نقاط الملاحظة.
- مساعدة حساسة للمحتوى.
- تحميل أجزاء أخرى.
- تنقيح نوافذ مجزأة.
- نافذة فورية.
- قشرة DOS
- أوامر أخرى من بيسك السريع.

يسمح لك بيسك السريع بكتابة برامج وتنفيذها وتصحيحها ويسمح لك بملفات رؤية وتتبع تنفيذ البرامج وتنقيح نماذج متعددة وعمل تجزئة لتنقيح الشاشة والتأكد من التكوين وقواعده أثناء ادخال البرنامج، ويقوم بيسك السريع بكتابة الكلمات المحجوزة بحروف كبيرة واتمام التنقيط تلقائياً لتسهيل ادخال البرنامج والتأكد من الأخطاء.

بدء بيسك السريع :

طريقة بدء بيسك السريع هي كما يلي :

١ - إذا كان لديك نظام قرص صلب تأكد أنك في دليل QB \ (أى أكتب CD\QB واضغط على مفتاح الادخال).

٢ - إذا كان لديك نظام به مشغلا أقراص مرنة، ادخل قرص البرنامج في المشغل A وادخل قرصاً مشكلاً وفارغاً في المشغل B، ثم انتقل للعمل على المشغل B.

٣ - إذا كنت تستخدم قرصاً صلباً اكتب QB من عند ملقن DOS واضغط على مفتاح الادخال. وإذا كنت تستخدم نظاماً به مشغلا أقراص مرنة اكتب A:QB واضغط على مفتاح الادخال.

لاحظ شاشة بيسك السريع التي تظهر، تسمح لك خيارات سطر الأمر بتوفير معلومات إضافية تؤثر على كيفية تنفيذ بيسك السريع، وخيارات سطر الأمر التالية يمكن أن تضاف عندما تبدأ بيسك السريع.

الخيار	الوصف
/run source file	تحميل ملف برنامج المصدر وتنفيذه، ملف المصدر source file هو اسم الملف الذي تريد تنفيذه.
/b	يستخدم موجهاً أبيض وأسود مع بطاقة رسومات ملونة، فإذا كانت لديك بطاقة لون واحد فلن تحتاج إلى استخدامه.
/g	للأجهزة التي بها رسومات ملونة ويقلل هذا الخيار من الترددات أثناء دوران النص.
/h	يستخدم أعلى ثبات ممكناً مع نظم المكونات الممكنة، فإذا كانت لديك بطاقة رسومات معززة وموجه رسومات معززة فيغير هذا الخيار العرض إلى 43 سطرًا و 80 عموداً.
/c: buffer size	يحدد حجم الذاكرة الاحتياطية للاتصالات، ولها قيمة تقليدية 512 بايت وأقصى حجم هو 32,767 بايت.
/1 library name	يحمل مكتبة Quick محددة.

الخيار	الوصف
/mbf	يتسبب في أن بيسك السريع يستخدم تشكيل ميكروسوفت الثنائي Microsoft Binary Format بدلاً من تشكيل اعداد IEEE.
/ah	يسمح بسجلات منظومات ديناميكية وسلاسل ثابتة الطول وبيانات عددية تكون أكبر من 64 كيلوبايت.
/cmd string	يستخدم سلسلة string لدالة \$COMMAND. ويجب أن يكون آخر عنصر في السطر.

كتابة برنامج

هذا القسم يقدم كيفية كتابة برنامج باستخدام منقح بيسك السريع. نفذ الأنشطة التالية :

- ١ - ابدأ بيسك السريع كما سبق ذكره في القسم السابق.
- ٢ - لاحظ وميض نقطة البداية في الركن العلوي الأيسر من الشاشة. وهذا يعني أن بيسك السريع ينتظر منك أن تفعل شيئاً.
- ٣ - ابدأ بكتابة البرنامج التالي. واضغط على مفتاح الادخال بعد نهاية كل سطر.

```

File Edit View Search Run Debug Calls File Help
<Untitled>
' This is a sample program
CLS
FOR cnt = 1 TO 10
  PRINT SPC(2 * cnt); "Allo thyar! "
NEXT cnt

Immediate
Main: <Untitled> Context: Program not running 00000:0.07
  
```

لاحظ أن cls و to و print و spc و next تصبح مكتوبة بحروف كبيرة بمجرد انتهائك من كتابة السطر والضغط على مفتاح الادخال. وهذا لأن منقح بيسك السريع ذكي ويميز كلمات بيسك السريع المحجوزة. والخطوات التالية توضح امكانيات أخرى لمنقح بيسك السريع الذكي.

٤ - استخدام مفاتيح نقطة البداية للانتقال إلى علامات التنصيص المزوجة المستخدمة في النهاية. (اضغط على Del لحذف علامة تنصيص النهاية).

لاحظ أن ببسك السريع يضيف علامة تنصيص مزوجة إلى السطر لإتمام التكوين.

٥ - الكلمة "cnt" موجودة في سطرين، انقل نقطة البداية إلى أقرب واحدة. غير الكلمة cnt إلى Cnt. انقل نقطة البداية إلى سطر آخر.

لاحظ أن ببسك السريع يغير الحوثر الآخر لـ cnt كذلك.

وفيما يلي قائمة بأوامر التنقيح المتاحة في ببسك السريع :

المفتاح	الاجراء
السهم الأيسر أو Ctrl-S	ينقل نقطة البداية خانة واحدة لليسار.
السهم الأيمن أو Ctrl-D	ينقل نقطة البداية خانة واحدة لليمين.
السهم الأيسر Ctrl-A أو Ctrl-	ينقل نقطة البداية كلمة واحدة لليسار.
السهم الأيمن Ctrl-F أو Ctrl-	ينقل نقطة البداية كلمة واحدة لليمين.
السهم العلوى أو Ctrl-E	ينقل نقطة البداية سطرأ واحداً لأعلى.
السهم السفلى أو Ctrl-X	ينقل نقطة البداية سطرأ واحداً لأسفل.
Ctrl-O-S أو Home	ينقل نقطة البداية لبداية السطر.
Ctrl-Q-D أو End	ينقل نقطة البداية لنهاية السطر.
Ctrl-J أو Ctrl-Enter	ينقل نقطة البداية لبدء سطر جديد.
Ctrl-R أو PgUp	ينقل نقطة البداية صفحة لأعلى.
Ctrl-C أو PgDn	ينقل نقطة البداية صفحة لأسفل.
Ctrl-W	يتحرك حركة دائمة لأعلى مع ثبات نقطة البداية في موقعها.
Ctrl-Z	يتحرك حركة دائمة لأسفل مع ثبات نقطة البداية في موقعها.
Ctrl-Q-E	ينقل نقطة البداية إلى قمة النافذة.
Ctrl-Q-X	ينقل نقطة البداية إلى قاعدة النافذة.

الاجراء	المفتاح
ينقل نقطة البداية لليسار بمقدار نافذة واحدة.	Ctrl-PgUp
ينقل نقطة البداية لليمين بمقدار نافذة واحدة.	Ctrl-PgDn
ينقل نقطة البداية إلى بداية جزء جديد أو بداية إجراء جديد.	Ctrl-Q-R أو Ctrl-Home
ينقل نقطة البداية إلى نهاية جزء جديد أو نهاية إجراء جديد.	Ctrl-Q-C أو Ctrl-End
وضع علامات (0-3).	Ctrl-K-number
الانتقال إلى علامات (0-3).	Ctrl-Q-number
تغيير حالة الادخال من on إلى off والعكس.	Ctrl-V أو Ins
ادخال سطر من أسفل.	End-Enter
ادخال سطر من أعلى.	Home-Ctrl-N
ادخال من لوحة القص.	Shift-Ins
عمل الجداول.	Tab
لطباعة Ctrl-any key.	Ctrl-P أو Ctrl-any key
لاختيار رمز على اليسار.	Shift- سهم أيسر
لاختيار رمز على اليمين.	Shift- سهم أيمن
لاختيار السطر الحالي.	Shift- سهم سفلى
لاختيار السطر الأعلى.	Shift- سهم علوى
لاختيار كلمة من على اليسار.	Shift-Ctrl-Left
لاختيار كلمة من على اليمين.	Shift-Ctrl-Right
لاختيار بقية الشاشة العلوية.	Shift-PgUp
لاختيار بقية الشاشة السفلية.	Shift-PgDn
لاختيار بداية جزء أو بداية أجزاء.	Shift-Ctrl-Home
لاختيار نهاية جزء أو نهاية أجزاء.	Shift-Ctrl-End
لحذف سطر أو حفظه فى لوحة القص.	Ctrl-Y
للحذف حتى نهاية السطر مع الحفظ فى لوحة القص.	Ctrl-Q-Y

حفظ البرنامج

يمكن أن يحفظ البرنامج الذى قمت بتنفيذه فى القسم السابق وذلك على قرص لاستخدامه فيما بعد، لعمل ذلك استمر على النحو التالى :

- ١ - اضغط على Alt، لاحظ أن اختيار القائمة الموجودة على أقصى اليسار يضاء.
- ٢ - اضغط على مفتاح السهم السفلى، لاحظ قائمة السحب لأسفل مع الخيارات المختلفة لمعالجة الملفات.
- ٣ - اكتب S لاختيار Save، يظهر صندوق يطلب ادخال اسم الملف.
- ٤ - اكتب Sample كاسم للملف، يدخل بيسك السريع التوسع BAS تلقائياً.

ملاحظة

إذا لم ترغب فى اضافة توسعات على أسماء الملفات اكتب نقطة بعد اسم الملف فوراً (مثل SAMPLE).

- ٥ - اضغط على Tab مرتين، لاحظ أن صندوق الحفظ يضاء.
 - ٦ - اضغط على قضيب المسافات، لاحظ أن الملف يحفظ.
- لقد سمحت فى الخطوة الخامسة بحفظ الملف كأحد ملفات بيسك السريع وهذا يسمح بتحميله بطريقة أسرع داخل المترجم. وهناك طريقة أخرى لحفظ الملف بحيث يمكن قراءته عن طريق برنامج آخر، وتجربة هذه الطريقة استمر كما يلى :
- ١ - اتبع الخطوات من 1 إلى 4 التى سبق ذكرها.
 - ٢ - اضغط على Tab ثم على مفتاح السهم السفلى، لاحظ أن الملف يذكر أنه يحفظ فى صورة ASCII.
 - ٣ - اضغط على Tab ثم اضغط على قضيب المسافات لحفظه.

استخدام صناديق الحوار

تحدث الحركة داخل كل صناديق الحوار dialog boxes (الصناديق التى تظهر عندما تجرى اختياراً من اختيارات القائمة وفى حالات أخرى عندما يتداخل بيسك السريع مع المستفيد) أساساً باستخدام المفاتيح التالية :

المفتاح	استخدامه
Tab	يأخذك من أحد الاختيارات الأساسية داخل صندوق الحوار إلى خيار آخر.
Shift-Tab	يعيدك إلى الخيار الذي مررت عليه دون أن تحدده بطريق الخطأ.
مفاتيح الأسهم	يساعدك على الحركة داخل الاختيار الذي حددته وتحديد مفاتيح أو اختيار حالات.
قضيب المسافات	يساعدك على وضع المفاتيح في حالة on أو off واختيار تنفيذ نشاط من صندوق الحوار.
مفتاح الإدخال	يستخدم للاستمرار في العملية مع مفاتيح وخيارات معينة. فإذا ما غيرت شيئاً واحداً فقط في صندوق الحوار فيمكنك إنهاء التداخل والاستمرار مع العملية دون أن تتحرك خلال كل خيارات هذا الصندوق الأخرى.

هذه المفاتيح وهذه المعلومات مقدمة هنا بحيث يمكنك أن تجربها وتعتاد على استخدام صناديق الحوار حيث إنها جزء متكامل من أجزاء بيئة تطوير بيسك السريع.

تحميل البرنامج

هذا القسم يتعامل مع كيفية تحميل ملف موجود في بيسك السريع.

١ - اضغط على Alt ثم على السهم السفلى.

٢ - أكتب O لفتح ملف. لاحظ صندوق Load مع سرد الدليل الحالي.

٣ - اضغط على Tab للانتقال إلى الدليل.

٤ - استخدم مفاتيح الأسهم للانتقال حول الدليل.

نصيحة : هناك طريقة أخرى لعمل ذلك وهي كتابة أول حرف من اسم الملف. ويأخذك هذا إلى أول اسم ملف يبدأ بهذا الحرف. مثال ذلك إذا كانت لديك ثلاثة ملفات تبدأ بنفس الحرف مثل

Factor. bas و Fun. bas و Func. bas فكتابة F أكثر من مرة واحدة تأخذك فى أول مرة إلى Factor. bas وفى المرة الثانية إلى Fun. bas وفى المرة الثالثة إلى Func. bas ثم تعيدك بعد ذلك إلى Factor. bas مرة أخرى. وهذه طريقة أسهل لاختيار الملفات إذا لم تكن متأكداً من الهجاء الصحيح لأسماء الملفات.

٥ - عندما تكون نقطة البداية على الملف SAMPLE. BAS اضغط على مفتاح الادخال.

كتابة برامج فرعية

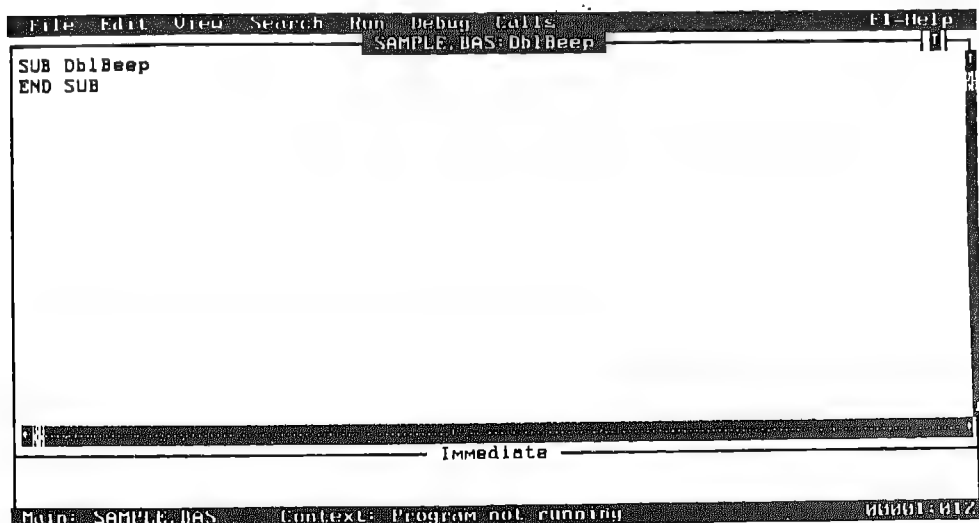
هذا القسم يوضح كيفية عزل بيسك السريع للبرامج الفرعية من الجزء المنادى وحفظها منظمة بحيث يمكنك نقل أحدها إلى الآخر أثناء التنقيح. ولحالة عمل ذلك استمر على النحر التالى :

١ - حمل البرنامج SAMPLE.BAS مستخدماً الاجراءات التى سبق ذكرها فى القسم السابق.

٢ - اضغط على Alt-E للذهاب إلى قائمة Edit.

٣ - أكتب S لاختيار SUB جديد. لاحظ النافذة التى تظهر سائلة عن اسم البرنامج الفرعى.

٤ - أكتب اسم البرنامج الفرعى Db1Beep واضغط على مفتاح الادخال.



٥ - لاحظ الشاشة الفارغة تحت عنوان DbIBeep SAMPLE.BAS ومجموعة SUB وENDSUB.

٦ - انقل نقطة البداية إلى نهاية أول سطر واضغط على مفتاح الإدخال. أكتب Beep: Beep.

الانتقال بين البرامج الفرعية

لتعريف البرنامج الفرعي DbIBeep داخل البرنامج SAMPLE.BAS استمر على النحو التالي :

١ - اضغط على Shift-F2 للذهاب إلى الجزء الرئيسي SAMPLE.BAS. يسمح لك Shift-F2 بالانتقال من برنامج فرعي لآخر.

٢ - اكتب DbIBeep قبل وبعد دورة FOR NEXT.

٣ - اضغط على Alt-F واكتب S لحفظ البرنامج. لاحظ عبارة DECLARE التي يدخلها بيسك السريع للبرنامج الفرعي الجديد DbIBeep. وتحدد الأقواس الفارغة أنه لا توجد هناك مؤشرات للبرنامج الفرعي DbIBeep.

٤ - اضغط على Shift-F5 لتنفيذ البرنامج. لاحظ الصوت المزدوج للصقارة قبل وبعد الطباعة.

لرؤية كيفية تناسق SUB مع البرنامج افعل ما يلي :

١ - انتقل إلى قائمة View. ويمكنك أن تفعل ذلك بأى من الطريقتين التاليتين :

- اضغط على Alt-F ومفتاح السهم الأيمن مرتين للانتقال إلى قائمة View

- أو اضغط على Alt-V.

٢ - اكتب S لـ Subs.

وهذا يعطى نافذة تعرض كل الأجزاء وكل البرامج الفرعية المحملة حالياً فى الذاكرة وكيفية ارتباطها ببعضها البعض. من هذه النافذة يمكنك أن تختار برنامجاً فرعياً أو جزءاً لتنقيحه أو لحذفه. اضغط على مفاتيح السهم العلوى والسفلى للانتقال من أحد المحتويات لآخر مع ملاحظة قاعدة الشاشة. يعرض اسم المحتوى مع تحديد ما إذا كان برنامجاً فرعياً أو جزءاً رئيسياً فإذا كان برنامجاً فرعياً فيظهر الجزء الذى ينتمى إليه هذا البرنامج الفرعى.

نصيحة : اضغط على F2 هو طريقة مختزلة للوصول إلى صندوق حوار SUB بدون

استخدام قائمة View.

القص واللصق

- هذا القسم يصف القص واللصق للأسطر داخل أحد الأجزاء. ابدأ عند صندوق حوار SUB الذى سبق شرحه فى القسم السابق واستمر على النحو التالى :
- ١ - انقل الاضاءة إلى DblBeep واضغط على مفتاح الإدخال ثم انقل نقطة البداية إلى العبارة .Beep: Beep
 - ٢ - اضغط على السهم السفلى Shift- ولاحظ السطر الذى يضاء.
 - ٣ - اضغط على Ctrl-Ins لحفظه فى لوحة القص. ولوحة القص هى مكان يخزن فيه النص لإجراء عملية لصق.
 - ٤ - انقل نقطة البداية إلى نهاية السطر واضغط على مفتاح الإدخال لإنتاج سطر جديد.
 - ٥ - اضغط على Shift-Ins للصق السطر من لوحة القص. لاحظ أن السطر يظهر.
 - ٦ - احفظ البرنامج كما سبق ذكره فى قسم سابق.
- يمكن أن تنفذ عملية القص واللصق كذلك على أجزاء من أسطر باستخدام سهم أيمن Shift- وسهم أيسر Shift-.

الإيجاد والحلال

- نتعلم من هذا القسم كيفية إيجاد سلسلة رموز فى البرنامج وحلال سلسلة أخرى محلها. كذلك تكون لديك الفرصة فى ممارسة التداخل مع صناديق الحوار. استمر كما يلى :
- ١ - اضغط على Shift-F2 للذهاب إلى الجزء الرئيسى. انقل بعد ذلك نقطة البداية إلى أول سطر فى البرنامج.
 - ٢ - اضغط على Ctrl-Q-A. يظهر صندوق لكتابة سلسلة البحث.
 - ٣ - اكتب "How Now Brown Cow" واضغط على Tab.
 - ٤ - تكون الآن فى الحقل الذى تريد أن تحدد فيه ما تريد أن تغيره How Now Brown Cow. اكتب "Don't Frown Beneath The Bough!" واضغط على Tab.
 - ٥ - انتقل إلى خيار Search واختر All Modules بالضغط على السهم السفلى ثم اضغط على Tab.

- ٦ - ضع البحث عند Match Upper/ Lowercase بالضغط على قضيب المسافات.
 - ٧ - اضغط على Tab مرتين لترك حقل Whole Word.
 - ٨ - انتقل إلى Find and Verify واستمر في العملية بالضغط على مفتاح الإدخال.
 - ٩ - عندما توجد السلسلة التي يجرى البحث عنها اسمح لها بالتغيير إلى السلسلة الجديدة وذلك بالضغط على مفتاح الإدخال كاستجابة لصناديق الحوار.
 - ١٠ - احفظ هذا البرنامج كما سبق ذكره في الأقسام السابقة.
- إذا كانت لديك أى مشاكل مع أى خطوة من الخطوات السابقة فيجب أن ترجع إلى قسم استخدام صناديق الحوار.

التتبع وتحديد نقاط ملاحظة

هذا القسم يقدم تمريناً في تتبع أحد البرامج. يسمح التتبع tracing بتنفيذ برنامج رؤية ما يحدث عندما ينفذ كل أمر من أوامره. وهذا مفيد عندما لا يعمل البرنامج ولا يكون لديك أى مفتاح لمعرفة ما يحدث فيه. ونقاط الملاحظة watch points هي شروط تقوم بوضعها بنفسك لمتغيرات توقف تنفيذ البرنامج عندما تتحقق هذه الشروط. وهذه وسيلة تصحيح أخرى في بيك السريع. جرب المثال التالى :

١ - حمل البرنامج SAMPLE.BAS إذا لم يكن محملاً بالفعل.

٢ - اضغط على Alt-D للذهاب إلى قائمة Debug واختر Watchpoint.

٣ - اكتب $Cnt = 5$ واضغط على مفتاح الإدخال. لاحظ أن نافذة الملاحظة تفتح عند قمة الشاشة مع اسم البرنامج والشروط المحدد.

SAMPLE.BAS: Cnt = 5

٤ - اضغط على Shift-F5 لتنفيذ البرنامج. لاحظ أن البرنامج يتوقف وأن الشرط في نافذة Watch هو <TRUE>. لاحظ كذلك سطر البرنامج المضاء عندما يتوقف التنفيذ.

٥ - اضغط على F4 لرؤية المخرجات. لاحظ أن أربعة أسطر فقط قد تم طباعتها. اضغط على F4 للعودة إلى نافذة الرؤية View.

٦ - احذف Watchpoint عن طريق اختيار Delete Watch من قائمة Debug.

يمكنك أن تدخل اسم متغير في نافذة Watch بدون شرط لرؤية القيم التى يفترضها المتغير أثناء تنفيذ البرنامج. استخدم خيار Add Watch من قائمة Debug واستخدم طريقة التنفيذ خطوة خطوة (F8). فيما يلي قائمة بالأوامر التى يمكنك استخدامها فى تتبع تنفيذ البرنامج.

المفتاح	الوصف
F4	الذهاب إلى شاشة المخرجات أو العودة إلى قائمة البرنامج.
F5	تنفيذ من عند العبارة الحالية.
Shift-F5	تنفيذ من البداية.
F7	تنفيذ البرنامج إلى موقع نقطة البداية.

المفتاح	الوصف
F8	تنفيذ عبارة البرنامج الحالية مع التتبع. أى تنفيذ كل عبارات البرنامج بما فى ذلك تنفيذ البرامج الفرعية والتتبع.
Shift-F8	التتبع التاريخى للخلف.
F9	تغيير نقاط التقاطع من on إلى off والعكس.
F10	تنفيذ العبارة التالية وتتبع الاجراء. أى تنفيذ كل عبارات البرنامج فى الجزء الرئيسى فقط وليس فى البرامج الفرعية أو تنفيذ التتبع.
Shift-F10	التتبع التاريخى للأمام.

مساعدة حساسة للمحتوى

المساعدة الحساسة للمحتوى context sensitive help هى مساعدة محددة لما تفعله عندما تطلب مساعدة، وفى بيسك السريع يمكنك الاتصال بها عن طريق الضغط على Shift-F1. وهى بسيطة جداً. استمر على النحو التالى :

١ - إذا لم تكن فى الجزء الرئيسى اضغط على F2 واختر SAMPLE.BAS من صندوق الحوار.

٢ - ضع نقطة البداية على عبارة FOR واضغط على Shift-F1.

لاحظ أن النصف العلوى للشاشة قد ظهر فيه توضيح لعبارة FOR.. NEXT. وهى قاعدة الشاشة ترى البرنامج الذى تم تنقيحه بواسطة عبارة FOR.

٣ - يمكنك الحركة حول البرنامج وطلب المساعدة على ذلك بنفس الطريقة. حاول عمل ذلك.

٤ - ضع نقطة البداية على استدعاء SUB لـ DbIBeep واطلب المساعدة. لاحظ أن بيسك السريع يفتح نافذة بكل كلمات بيسك السريع المحجوزة ويسالك أن تختار واحدة منها. وهذا لأن DbIBeep ليست كلمة بيسك سريع محجوزة. وعلى هذا لا توجد مساعدة متاحة لها وعند ذلك يفعل بيسك السريع الشيء الجيد التالى ويسمح لك باختيار أحد الأشياء. لاختيار واحدة ضع الاضاعة عليها مستخدماً مفاتيح الأسهم واضغط على مفتاح الادخال.

٥ - اضغط على Esc مرتين للعودة إلى قائمة البرنامج.

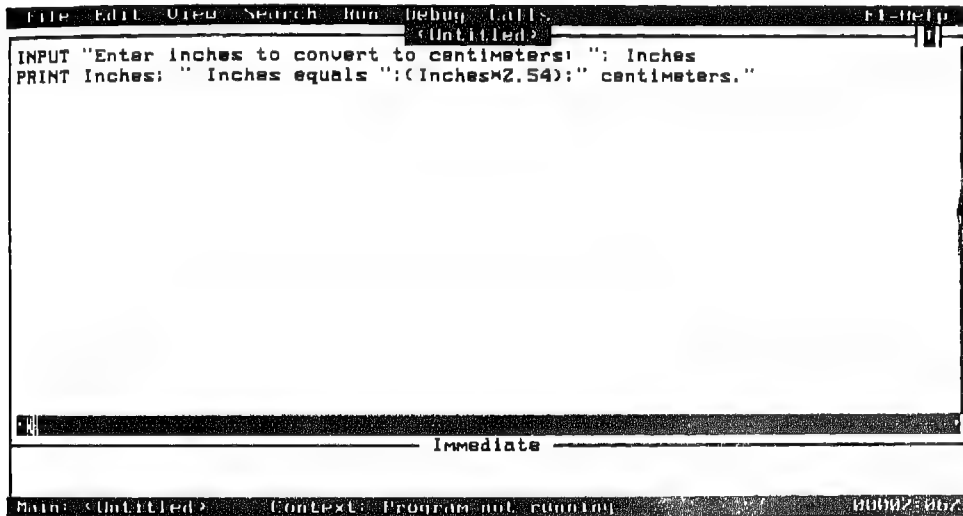
تحميل أجزاء أخرى

فى بيسك السريع الجزء module عبارة عن مجموعة من أسطر البرنامج محفوظة فى ملف منفصل يمكن تنفيذه بنفسه أو مع أجزاء أخرى.

وفى هذا القسم تقوم بكتابة برنامج صغير آخر ثم تحمل البرنامج SAMPLE.BAS فى بيسك السريع دون أن تفقد البرنامج الجديد. استمر على النحو التالى :

١ - أحفظ البرنامج SAMPLE.BAS واختر New من قائمة File بالضغط على Alt-F وكتابة N.

٢ - اكتب البرنامج التالى :



The screenshot shows a BASIC interpreter window with a menu bar (File, Edit, View, Search, Run, Debug, Calls, F1-Help) and a title bar (Untitled). The main text area contains the following code:

```
INPUT "Enter inches to convert to centimeters: "; Inches
PRINT Inches; " Inches equals ";(Inches*2.54);" centimeters."
```

Below the text area is an "Immediate" window. At the bottom, a status bar shows "Main: <Untitled>" and "Context: Program not running".

٣ - احفظ هذا البرنامج على أنه ملف ASCII تحت اسم INCH2CM.BAS.

٤ - انتقل إلى قائمة File واختر Load, لاحظ صندوق حوار تحميل الملف مع خيارات الحالة التى تريدها وقائمة الدليل, اجعل الحالة على انها جزء module.

٥ - انتقل إلى الدليل بالضغط على Tab مرتين واختر SAMPLE.BAS واضغط على مفتاح الإدخال.

٦ - انتقل إلى قائمة View واكتب S لرؤية كل الأجزاء وكل البرامج الفرعية.

نصيحة : F2 هي طريقة مختزلة لأداء نفس الشيء.

٧ - باستخدام مفاتيح الأسهم اختر SAMPLE.BAS ولاحظ الوصف الموجود في القاعدة.

هذا يبين أنك تستطيع تحميل أكثر من ملف واحد في الذاكرة في نفس الوقت، وحيث إن الجزئين المحملين ليسا مرتبطين مع بعضهما البعض فينفذ أول نموذج تم تحميله فقط عندما تضغط على Shift-F5، ولتنفيذ أى نموذج آخر تم تحميله حدد أن الجزء الرئيسى هو هذا الجزء وذلك من قائمة Run، ولحاولة ذلك افعل ما يلى :

١ - اضغط على esc للعودة إلى قائمة البرنامج.

٢ - انتقل إلى قائمة Run واختر Set Main Module. لاحظ ظهور صندوق الحوار مع قائمة بكل الأجزاء المحملة.

٣ - باستخدام مفاتيح الأسهم اختر INCH2CM.BAS ثم اضغط على Tab ثم على قضيب المسافات.

٤ - اضغط على Shift-F5 لتنفيذ الجزء الرئيسى الجديد مع ملاحظة أن SAMPLE.BAS لم ينفذ.

٥ - نفذ الخطوات من 1 إلى 3 وحدد أن SAMPLE.BAS هو الجزء الرئيسى.

تنقيح الشاشة المجزأة

ترى في هذا القسم برنامجين أو جزئين من نفس البرنامج وذلك في نفس الوقت، استمر على النحو التالى :

١ - انتقل إلى قائمة File واختر New. يلغى هذا الخيار كل البرامج الموجودة في الذاكرة بحيث يمكنك أن تكتب برنامجاً جديداً. لاحظ أن ببسك السريع يلقنك بأن الأجزاء المحملة حالياً في الذاكرة لم تحفظ. اختر No فى صندوق الحوار ثم اضغط على قضيب المسافات.

٢ - انتقل إلى قائمة File واختر Open. حمل البرنامج SAMPLE.BAS.

٣ - اختر قائمة View وأكتب P لترى ثلاث نوافذ على الشاشة. لاحظ الثلاث نوافذ، الأولى منها الجزء الرئيسى والثانية معها نفس الجزء والنافذة الفورية.

٤ - اضغط على F6 للانتقال إلى النافذة التالية. والآن اضغط على Shift-F2 لرؤية البرنامج الفرعى Db1Beep. لاحظ أنك تنتظر إلى جزعين مختلفين من نفس البرنامج فى نفس الوقت.

```

File Edit View Search Run Debug Calls F1=Help
SAMPLE.BAS
DECLARE SUB Db1Beep ( )
'This is a sample program
CLS
Db1Beep
FOR Cnt = 1 TO 10
  PRINT SPC(2 * Cnt); "Hello there! "
NEXT Cnt
Db1Beep

SAMPLE.BAS
DECLARE SUB Db1Beep ( )
'This is a sample program
CLS
Db1Beep
FOR Cnt = 1 TO 10
  PRINT SPC(2 * Cnt); "Hello there! "
NEXT Cnt
Db1Beep

Immediate

Main: SAMPLE.BAS Context: Program not running 00001:001
  
```

٥ - اضغط على Shift-F5 للعودة إلى الخلف نافذة واحدة. اضغط على F6 يأخذك من نافذة إلى أخرى وذلك من النافذة العلوية إلى النافذة السفلية. ويأخذك الضغط على Shift-F6 من النافذة السفلية إلى النافذة العلوية واحدة بعد الأخرى.

٦ - اضغط على Ctrl-F10 لإعادة النافذة الحالية إلى شاشة كاملة، يمكن عمل ذلك مع أى من الثلاث نوافذ.

٧ - اختر New من قائمة File. استجب كما فعلت فى الخطوة رقم 1 للقرن بيسك السريع.

النافذة الفورية

النافذة الفورية immediate window هى سمة تسمح لك بادخال عبارات برنامج بيسك وتنفيذها بالضغط على مفتاح الادخال. حاول عمل ذلك باتباع الخطوات التالية :

١ - اضغط على F6 للانتقال إلى النافذة الفورية Immediate Window.

٢ - اكتب العبارة التالية :

Cls: For i = 1 to 10: Print "Ze emmediate weendow"

٣ - اضغط على مفتاح الادخال لتنفيذ عبارة البرنامج، لاحظ أن العبارة تنفذ على الفور.

وفيما يلي خواص النافذة الفورية :

- يمكنك أن تنفذ سطرأً فردياً بوضع نقطة البداية على السطر والضغط على مفتاح الادخال، ولاينفذ إلا هذا السطر.
- يمكنك ادخال عدد من السطور لا يزيد عن 10 أسطر ولاتزيد الرموز في السطر الواحد عن 256 رمزاً. يمكنك كتابة أكثر من عبارة واحدة على نفس السطر. أنظر المثال السابق.
- يمكنك أن تغير الحجم بالضغط على Alt++ للزيادة أو Alt-- للنقصان أو Ctrl-F10 للعودة إلى الشاشة الكاملة.
- يمكن حفظ الشفرة على قرص إلا إذا استخدمت أوامر القص والنسخ Cut and Copy من قائمة Edit.
- يمكن استخدام معظم الدوال والعبارات في النافذة الفورية. وفيما يلي قائمة بكلمات بيسك السريع المحجوزة غير المسموح بها في النافذة الفورية.

COMMON	DEF FN	SHARED	\$DYNAMIC
DATA	DIM	STATIC	REDIM
\$STATIC	OPTION	TYPE	END IF
SUB	END SUB	END DEF	END TYPE
ELSEIF	FUNCTION	DECLARE	\$INCLUDE
END FUNCTION	CONST	DEFtype	

وفيما يلي بعض استخدامات النافذة الفورية :

- اختبار عبارات PRINT.

- اختبار الحسابات.
- استدعاءات إجراءات لاختبارها منفصلة.
- تغيير قيم المتغيرات في تنفيذ البرنامج. ولعمل ذلك نفذ البرنامج من نافذة View وأفضله باستخدام Ctrl-Break وانتقل إلى النافذة الفورية وغير قيمة المتغير وأضغط على F5 للاستمرار في التنفيذ. والسطر المضاء في نافذة View هو السطر الذي يتأثر.
- محاكاة أخطاء وقت التنفيذ بتحديد قيمة خطأ لعبارة ERROR.
- ٤ - اضغط على F6 للعودة إلى نافذة View.

قشرة DOS

- هذا القسم يصف اختيار DOS Shell من قائمة File. ويسمح DOS Shell بترك بيسك السريع لحظيا والذهاب لتنفيذ أوامر DOS ثم العودة إلى بيسك السريع دون فقدان بيانات ولمحاولة ذلك استمر على النحو التالي :
- ١ - انتقل إلى قائمة File واختر DOS Shell. تظهر التعليمات الخاصة بالعودة إلى بيسك السريع وملقن DOS.

```

Ze immediate weendow
Ze immediate weendow
Ze immediate weendow
Ze immediate weendow
Ze immediate weendow
Ze immediate weendow
Ze immediate weendow
Ze immediate weendow
Ze immediate weendow
Ze immediate weendow
Ze immediate weendow
Type EXIT to return to QuickBASIC

The IBM Personal Computer DOS
Version 3.3B (C)Copyright International Business Machines Corp 1981, 1987
(C)Copyright Microsoft Corp 1981, 1986

C:\QB>
```

٢ - اكتب DIR/W عند ملقن DOS واضغط على مفتاح الإدخال. لاحظ أن الدليل تظهر محتوياته.

٣ - اكتب EXIT واضغط على مفتاح الإدخال. لاحظ ظهور العبارة المعتادة "Press any key to continue" من بيسك السريع. افعل ذلك للعودة إلى بيسك السريع.

٤ - اضغط على Alt-F وأكتب X للخروج من بيسك السريع.

أوامر بيسك سريع أخرى

فيما يلي أوامر أخرى متاحة في بيسك السريع :

المفتاح	الغرض منه
F1	يعطى معلومات مساعدة عامة.
F2	يسرد الملفات المحملة (SUB و FUNCTION و module و include و document).
F3	يعيد آخر أمر ايجاد.
F4	يبين شاشة المخرجات.
F5	يستمر مع تنفيذ البرنامج.
F6	ينقل إلى النافذة التالية.
F7	ينفذ البرنامج إلى موقع نقطة البداية.
F8	ينفذ العبارة التالية مع التتبع خلال الاجراءات.
F9	ينقل من وإلى نقاط التقاطع.
F10	ينفذ العبارة التالية مع التتبع خلال الإجراء.
Shift-F6	يعيد إلى النافذة السابقة.
Ctrl-F2	يعيد إلى البرنامج السابق.
Ctrl-F5	يعيد الشاشة الكاملة إلى حجمها السابق.
Ctrl-F10	يجعل النافذة النشطة كاملة الحجم.
Ctrl-\	يحضر النص المختار.

المفتاح	الفرضونه
Ctrl-Q-A	يعيد الایجاد والاستبدال.
Alt-Bksp	يلغى آخر تنقيح.
Esc	يفلق آخر نافذة.

انتقل إلى الدرس الثالث للاستمرار في تسلسل التعلم.

ملحق (C)

جدول ASCII وكلمات بيسك السريع المحبوزة

جدول ASCII

Regular ASCII Chart (character codes 0 - 127)															
000	(nul)	016	(dle)	032	sp	048	0	064	0	080	P	096	`	112	p
001	(soh)	017	(dc1)	033	!	049	1	065	A	081	Q	097	a	113	q
002	(stx)	018	(dc2)	034	"	050	2	066	B	082	R	098	b	114	r
003	(etx)	019	(dc3)	035	#	051	3	067	C	083	S	099	c	115	s
004	(eot)	020	(dc4)	036	\$	052	4	068	D	084	T	100	d	116	t
005	(enq)	021	(nak)	037	%	053	5	069	E	085	U	101	e	117	u
006	(ack)	022	(syn)	038	&	054	6	070	F	086	V	102	f	118	v
007	(bel)	023	(etb)	039	'	055	7	071	G	087	W	103	g	119	w
008	(bs)	024	(can)	040	(056	8	072	H	088	X	104	h	120	x
009	(tab)	025	(em)	041)	057	9	073	I	089	Y	105	i	121	y
010	(lf)	026	(eof)	042	*	058	:	074	J	090	Z	106	j	122	z
011	(vt)	027	(esc)	043	+	059	;	075	K	091	[107	k	123	{
012	(fp)	028	(fs)	044	,	060	<	076	L	092	\	108	l	124	
013	(cr)	029	(gs)	045	-	061	=	077	M	093]	109	m	125	}
014	(so)	030	(rs)	046	.	062	>	078	N	094	^	110	n	126	~
015	(si)	031	(us)	047	/	063	?	079	O	095	_	111	o	127	Δ

Extended ASCII Chart (character codes 128 - 255)															
128	€	144	€	160	€	176	€	192	€	208	€	224	€	240	€
129	€	145	€	161	€	177	€	193	€	209	€	225	€	241	€
130	€	146	€	162	€	178	€	194	€	210	€	226	€	242	€
131	€	147	€	163	€	179	€	195	€	211	€	227	€	243	€
132	€	148	€	164	€	180	€	196	€	212	€	228	€	244	€
133	€	149	€	165	€	181	€	197	€	213	€	229	€	245	€
134	€	150	€	166	€	182	€	198	€	214	€	230	€	246	€
135	€	151	€	167	€	183	€	199	€	215	€	231	€	247	€
136	€	152	€	168	€	184	€	200	€	216	€	232	€	248	€
137	€	153	€	169	€	185	€	201	€	217	€	233	€	249	€
138	€	154	€	170	€	186	€	202	€	218	€	234	€	250	€
139	€	155	€	171	€	187	€	203	€	219	€	235	€	251	€
140	€	156	€	172	€	188	€	204	€	220	€	236	€	252	€
141	€	157	€	173	€	189	€	205	€	221	€	237	€	253	€
142	€	158	€	174	€	190	€	206	€	222	€	238	€	254	€
143	€	159	€	175	€	191	€	207	€	223	€	239	€	255	€

كلمات بيسك السريع المحجوزة

الكلمات المحجوزة هي كلمات تحجز كقوائم للغة البرمجة. ولا يمكن استخدام هذه الكلمات في أى شيء آخر غير الغرض الذى سبق تحديده لها. وتسمى الكلمات المحجوزة reserved words بأنها كلمات رئيسية key words أو بأنها أفعال اللغة language verbs كذلك. وفيما يلى قائمة بكلمات بيسك السريع المحجوزة.

ABS	DECLARE	INSTR	OUT	SPACES
ACCESS	DEF	INT	OUTPUT	SPC
ALIAS	DEFDBL	INTEGER	PAINT	SQR
AND	DEFINT	IOCTL	PALETTE	STATIC
ANY	DEFLNG	IOCTLS	PCOPY	STEP
APPEND	DEFSNG	IS	PEEK	STICK
AS	DEFSTR	KEY	PEN	STOP
ASC	DIM	KILL	PLAY	STRS
ATN	DO	LBOUND	PMAP	STRING
BASE	DOUBLE	LCASES	POINT	STRING
BEEP	DRAW	LEFTS	POKE	STRINGS
BINARY	ELSE	LEN	POS	SUB
BLOAD	ELSEIF	LET	PRESET	SWAP
BSAVE	END	LINE	PRINT	SYSTEM
BYVAL	ENDIF	LIST	PSET	TAB
CALL	ENVIRON	LOC	PUT	TAN
CALLS	ENVIRON\$	LOCAL	RANDOM	THEN
CASE	EOF	LOCATE	RANDOMIZE	TIMES
CDBL	EQV	LOCK	READ	TIMER
CDECL	ERASE	LOF	REDIM	TO
CHAIN	ERDEV	LOG	REM	TROFF
CHDIR	ERDEVS	LONG	RESET	TRON
CHRS	ERL	LOOP	RESTORE	TYPE
CINT	ERR	LPOS	RESUME	UBOUND
CIRCLE	ERROR	LPRINT	RETURN	UCASES
CLEAR	EXIT	LSET	RIGHTS	UNLOCK
CLNG	EXP	LTRIMS	RMDIR	UNTIL
CLOSE	FIELD	MID\$	RND	USING
CLS	FILEATTR	MKD\$	RSET	VAL
COLOR	FILES	MKDIR	RTRIM\$	VARPTR
COM	FIX	MKDMBF\$	RUN	VARPTR\$
COMMANDS	FOR	MKIS	SADD	VARSEG
COMMON	FRE	MKL\$	SCREEN	VIEW
CONST	FREEFILE	MKS\$	SEEK	WAIT
COS	FUNCTION	MKSMBF\$	SEG	WEND
CSNG	GET	MOD	SELECT	WHILE
CSRLIN	GOSUB	NAME	SETMEM	WIDTH
CVD	GOTO	NEXT	SGN	WINDOW
CVDMBF	HEX\$	NOT	SHARED	WRITE
CVI	IF	OCT\$	SHELL	XOR
CVL	IMP	OFF	SIGNAL	
CVS	INKEY\$	ON	SIN	
CVSMBF	INP	OPEN	SINGLE	
DATA	INPUT	OPTION	SLEEP	
DATE\$	INPUT\$	OR	SOUND	

ملحق (D)

مترجم وواصل سطر الأوامر

يمكن أن تترجم البرامج التي كتبتها باستخدام بيسك السريع بحيث يمكنك تنفيذها من ملقن DOS بدون استخدام بيسك السريع. إحدى طرق عمل ذلك هي إنتاج ملف EXE. من بيئة بيسك السريع. الطريقة الأخرى هي استخدام مترجم وواصل سطر أوامر البيسك. ويصف هذا الملحق طريقة ترجمة سطر الأوامر. وفيما يلي قائمة بالملفات المستخدمة في هذه الطريقة مع وصف موجز لها.

الملف	الوصف
BC.EXE	هذا هو مترجم سطر الأوامر ويستخدم عند ترجمة البرامج المكتوبة بمنقح آخر أو ترجمة برامج أكبر من أن تتم ترجمتها داخل الذاكرة.
LINK.EXE	هذا هو واصل Microsoft Overlay Linker المستخدم في توصيل ملفات Obj. مع مكتبات وقت التنفيذ اللازمة.

الترجمة بواسطة BC. EXE

هذا القسم يصف كيفية استخدام برنامج BC.EXE في ترجمة ملفات المصدر. وتكوين مترجم سطر الأوامر BC هو كما يلي :

```
BC srcfile,objfile,listfile optionlist;
```

جزء srcfile هو ملف مصدر بلغة البيسك. وجزء objfile هو اسم الملف الذي تذهب إليه الشفرة التنفيذية وعندما يحذف هذا الجزء فيكون اسم ملف التنفيذ هو نفس اسم ملف المصدر ولكن باتساع Obj.. وجزء listfile هو اسم الملف الذي تذهب إليه قائمة برنامج المصدر وعندما يحذف هذا الجزء فلا يتم إنتاج ملف بقائمة المصدر. وجزء optionlist هو قائمة بخيارات ترجمة

تكون مهمة عند استخدام بعض سمات بيسك السريع، وفيما يلي قائمة خيارات سطر الأمر BC.EXE المختلفة :

الخيار	الوصف
/a	لانتاج شفرة أشياء غير مجمعة وتوضيح شفرة التجميع التي تنتج لكل سطر من أسطر البرنامج.
/ah	يسمح بمنظومات ديناميكية لسجلات وبسلاسل ثابتة الطول وأنواع عديدة يكون حجمها أكبر من 64 كيلوبايت.
/c:bufsize	تحدد حجم الذاكرة الاحتياطية للاتصالات وحدها الأقصى هو 32,767 وقيمتها التقليدية هي 256 بايت. وهي ذاكرة احتياطية للاستقبال فقط.
/d	تنتج شفرة تصحيح للتأكد من أخطاء وقت التنفيذ وللتمكن من عمل Ctrl-Break.
/e	تستخدم في البرامج مع عبارات ON ERROR و RESUME وتحدد وجود هذه العبارات.
/mbf	تعالج الأعداد في تشكيل IEEE على أنها أعداد لها شكل ميكروسوفت الثنائي. تعالج الدوال CVD و CVS و MKD\$ و MKS\$ كدوال CVDMBF و CVSMBF و MKDMBF\$ و MKSMBF\$.
/o	تعرض بـ BCOM40. LIB بدلاً من BRUN40. LIB. وتستخدم عندما يترجم البرنامج بحيث لا يكون وقت التنفيذ الذي يدعم مكتبة BRUN40. BAS مطلوباً. ويكون البرنامج المترجم أطول.
/r	تخزن صف منظومة رئيسياً وتستخدم مع مقاطع أخرى مكتوبة بلغات أخرى إذا كانت تخزن صف منظومة رئيسياً.
/s	تكتب سلسلة ثابتة في ملف التنفيذ بدلاً من جدول الرموز. تستخدم مع البرامج التي تستخدم كميات من ثوابت السلاسل.

الخيار	الوصف
/v	تمكن من اصطياد الأحداث وتستخدم مع عبارة ON event.
/w	نفس الشيء مثل /v مع التأكد من الأحداث بين العبارات.
/x	تحدد أن البرنامج يحتوى على عبارات ON ERROR و RE-SUME و RESUME NEXT و RESUMED.
/zd	تنتج ملف تنفيذ بسجلات لعدد الأسطر تناظر ملف المصدر. تستخدم عندما ترغب فى استخدام معيد التصحيح الرمزي لميكروسوفت Microsoft Symbolic Debugger (SYMDEB).
/zi	تنتج شفرة تنفيذ متوافقة مع معيد تصحيح شفرة رؤية ميكروسوفت Microsoft Code View.

التوصيل : هذا القسم يصف استخدام واصل ميكروسوفت Microsoft Overlay Linker (LINK.EXE). ويمكن استدعاء برنامج الوصل بإحدى الطرق التالية :

- كتابة عبارة LINK باستخدام التكوين التالى :

```
LINK objfile,exefile,mapfile,lib linkopts;
```

حيث objfile هو الملف الذى ينتج بواسطة BC.EXE وجزء exefile هو المكان الذى يذهب إليه البرنامج الذى يتم توصيله (وعندما يحذف هذا الجزء يكون للملف نفس الاسم مثل ملف obj. ولكن يكون اتساعه هو .exe). وجزء mapfile يرسم قائمة إذا ما اخترته أما جزء lib, فهو مكتبة واحدة أو أكثر يراد ايصالها بملف التنفيذ لانتاج ملف .exe. أما جزء linkopts فهو قائمة خيارات الاتصال.

- كتابة LINK والضغط على مفتاح الادخال والاجابة عن الأسئلة.

- اعداد ملف بالاستجابات المناسبة بنفس الترتيب مثل الأسئلة وكتابة Link & responsefile عند ملقن DOS حيث responsfile هو اسم الملف للاستجابات التى انتجته.

وفيما يلي قائمة بخيارات التوصيل المختلفة التي يقبلها LINK. EXE

الخيار	الوصف
/HE	رؤية قائمة الخيارات المتاحة.
/PAU	تستخدم عندما تكون المقايضة للفرص مشمولة وتتسبب في إيقاف التوصيل قبل كتابة ملف التنفيذ على القرص.
/I	تعرض معلومات عن عملية التوصيل.
/B	تتسبب في أن الواصل لا يعطى ملقناً إذا لم يستطع أن يجد مكتبة أو ملف تنفيذ. ويحدث التوصيل في صورة تشغيل دفعة.
/Q	تدمج ملفات التنفيذ المحددة لمكتبة سريعة. يجب أن يدعم BQLB40.LIB كأحد مكتباته.
/E	تحذف تسلسل بايت مكررة ويجعل الشفرة أكثر إيجازاً لجعل التحميل أكثر سرعة.
/NOP	تفلق عمل تعبئة قطاع الشفرة.
/NOD	لا يتم البحث في المكتبات التي تأتي بعده لحل resolve الاختلافات الخارجية. يستخدم لتجنب تحميل مكتبات نمطية مكررة.
/SE	للتحكم في عدد القطاعات التي يسمح الواصل بها للبرنامج. القيمة التقليدية هي 128 ويمكن أن تصل إلى 1024 كحد أقصى.
/M	إنتاج ملف خريطة.
/LI	تشمل أرقام السطور والعناوين المصاحبة لبرنامج المصدر في ملف الخريطة.
/PAC: number	تتسبب في أن الواصل يجمع قطاعات الشفرة المتجاورة بحيث أنها تقتسم نفس عنوان القطاع. والرقم المحدد هو حجم المجموعة وله قيمة تقليدية 65,530.
/CO	تنتج ملف Exe. يمكن تصحيحه باستخدام Code View De-bugger ويجب أن يستخدم خيار /zi أثناء الترجمة.

الوصف	الخيار
تتسبب في أن الواصل يميز بين الحروف الكبيرة والحروف الصغيرة، لا تستخدمه.	/NOI
تضع أقصى عدد للمقاطع (16 بايت) يتطلبه البرنامج عند تحميله، ويراقب الذاكرة أثناء تنفيذ البرنامج.	/CP
تجبر ترتيب القطاعات طبقاً لمنتجات لغة Microsoft مرتفعة المستوى التقليدية.	/DO
تضع حجم للرصة ويقع بين 1 و 65,535 وله قيمة تقليدية 2 كيلوبايت.	/ST: number
تحميل كل البيانات بدءاً عند النهاية المرتفعة لقطاع البيانات التقليدي.	/DS
تضع البرامج مرتفعة بقدر الإمكان في الذاكرة.	/HI
تهمل مصاحبات المجموعات عند تحديد عناوين للبيانات وللشفرة.	/NOG
تحدد رقم ازعاج مختلفاً من 0x3F عند تمرير التحكم إلى الغطاء overlay.	/O

ملحق E

المكتبات

مقدمة

هذا الملحق يناقش استخدام المكتبات مع بيسك السريع، والمكتبات هي دوال وإجراءات يكتبها المستفيدون يكون قد تم اختبارها كما أنها تكون مستقلة في تنفيذها أى إنها لا تستخدم أى متغيرات أو موارد تكون جزءاً من برنامج آخر، ويسمح لك بيسك السريع بتحميل أحد أنواع المكتبات يسمى بالمكتبات السريعة Quick library ويكون للملف الاتساع QLB.. وعندما تحمل مثل هذه المكتبة أثناء استدعاء بيسك السريع فتصبح أجزاء المكتبة اتساعاً لمكتبة وقت تنفيذ بيسك السريع وتصبح البرامج الفرعية متاحة بدون توضيحات صريحة. والنوع الآخر للمكتبة هو نوع يستخدم في مرحلة توصيل لترجمة ملف EXE، وهذه المكتبة لها اتساع LIB.. عندما ينتج بيسك السريع مكتبة سريعة فإنه ينتج صيغة LIB، للمكتبة لتوصيلها مع البرنامج، ويصف هذا الملحق الأنشطة التالية :

- تحميل المكتبة السريعة.
- انتاج مكتبة سريعة.
- رؤية محتويات مكتبة سريعة.
- استخدام مدير مكتبة قائم بذاته.

تحميل مكتبة سريعة

لتحميل مكتبة سريعة quick عند تشغيل بيسك السريع فإنك تبدأ عند ملقن DOS وتكتب QB/L واسم المكتبة السريعة المطلوب وتضغط على مفتاح الادخال. مثال ذلك لتحميل مكتبة سريعة اكتب QB/LQB واضغط على مفتاح الادخال. تأكد أن المكتبة السريعة المحددة تكون متاحة من أعداد مسار DOS الحالي.

انتاج مكتبة سريعة

لتعلم كيفية انتاج مكتبة سريعة من بيئة بيسك السريع حاول أن تؤدي الأنشطة التالية :

- ١ - نفذ بيسك السريع وانتج برنامجاً فرعياً جديداً تحت اسم InchToCm.

٢ - اكتب أسطر البرنامج التالية فى البرنامج الفرعى.

```
SUB InchToCm
  INPUT "Enter inches to convert to centimeters: "; Inches
  PRINT Inches; " inches is equal to "; (Inches * 2.54); " centimeters"
END SUB
```

٣ - أختَر Make Library من قائمة Run واختر Yes فى صندوق الحوار المرحلى فاحصاً إذا ما كنت تريد حفظ الملف أم لا. اكتب Test كاسم مكتبة سريعة واختر Product Debug Code واختر Make Library.

٤ - لاحظ أسطر الأوامر التى ينتجها بيسك السريع مع انتاجه لمكتبة سريعة جديدة.

٥ - اخرج من بيسك السريع. نفذ بيسك السريع مرة أخرى وفى هذه المرة حمل المكتبة السريعة التى انتجتها.

٦ - انتقل إلى النافذة الفورية واكتب Call InchToCm ولاحظ أن البرنامج الفرعى ينفذ.

رؤية محتويات المكتبة السريعة

هناك برنامج مقدم كمثال فى قرص توزيع بيسك السريع اسمه QLBDUMP.BAS. يعطيك هذا البرنامج قائمة بمحتويات مكتبة سريعة محددة. ونستعرض فى هذا القسم محتويات المكتبة السريعة التى سبق انتاجها فى القسم السابق.

١ - حمل بيسك السريع وحمل البرنامج QLBDUMP.BAS ونفذه.

٢ - اكتب Test عند الملحق كاسم للمكتبة. مخرجات هذا البرنامج هى كما يلى :

```
Code Symbols:
  _brkctl
  _execve
  _exit
  _main
  _spawnve
  INCHTOCM

Data Symbols:
  b_erradr
  b_ULVers
  b_errlin
  b_errmod
  _and
  b_errnum
  b_ULSymSeg
  STXH00
  _environ
  _errno
  _odata

Press any key to continue
```

لاحظ اسم البرنامج الفرعى فى النصف الأول من المخرجات.

استخدام مدير مكتبة سريعة قائم بذاته

مدير المكتبة السريعة القائم بذاته عبارة عن برنامج مقدم على أنه ملف EXE. يسمح لك بإضافة برامج فرعية أو حذفها أو نقلها أو استبدالها أو نسخها أو دمجها من مكتبة LIB. لتكوين مكتبة جديدة. واسم البرنامج هو LIB.EXE. وفيما يلي قائمة بالعمليات المختلفة التي يمكن أن ينفذها LIB.EXE على مكتبة معينة.

الوصف	الأمر
يضيف المكتبة أو ملف Obj. للمكتبة.	+ objfill / lib
يحذف جزءاً معيناً من المكتبة.	- module
يستبدل جزء محدد.	- + module
ينسخ جزءاً محدداً في ملف Obj..	* module
ينقل جزءاً محدداً إلى ملف Obj..	- * module

يمكن استدعاء برنامج LIB بإحدى الطرق التالية :

- كتابة ما يلي عند ملقن DOS.

```
LIB oldlib /P:number Commands,listfile,newlib;
```

- حيث oldlib هو اسم المكتبة التي تغيرها أو تنتجها. (فإذا لم يوجد الاسم فيعطيك البرنامج فرصة لإنتاج واحدة). و /P:number هو حجم الصفحة للمكتبة وهو قوة 2 بين 16 و 32,768. يحدد هذا الأمر نوع العمليات التي تريد تنفيذها على المكتبة. وهي مذكورة في الخريطة السابقة. و Listfile هو اسم ملف قائمة تقاطع الأدلة. و Newlib هو اسم المكتبة المعدلة. ويعطى اتساع BAK. لاسم المكتبة القديم إذا لم تحدد اسم مكتبة جديداً.

- اكتب LIB مع الإجابة على الأسئلة، والمدخلات تشبه جداً صيغة سطر الأوامر فيما عدى مواصفة حجم الصفحة. ولإعطاء ملفات أكثر لأى ملقن أكتب & عند نهاية السطر ويعيد البرنامج السؤال.

- قم باعداد ملف بالاستجابات المناسبة بنفس ترتيب الأسئلة مع كتابة، عند ملقن DOS،
مايلي : LIB & responsefile حيث responsefile هو الملف الموجود فيه الاستجابات والذي
سبق أن أعدته.

ملحق F

رسائل الخطأ

أخطاء الاستدعاء والترجمة ووقت التنفيذ

هذا الملحق يصف رسائل الخطأ التي ينتجها بيسك السريع سواء كان ذلك أثناء استدعاء بيسك السريع أو أثناء ترجمة أو تنفيذ البرنامج. والجدول التالي يسرد رموز أخطاء وقت التنفيذ والرسائل المصاحبة لها.

الرمز	الرسالة
3	العودة بدون GOSUB.
4	لا توجد بيانات.
5	استدعاء دالة غير صحيح.
6	سريان زائد.
7	الذاكرة غير كافية.
9	دليل المنظومة يقع خارج المدى.
11	قسمة على صفر.
14	مكان السلسلة لا يكفي.
16	صيغة السلسلة معقدة جداً.
19	لا يوجد RESUME.
20	يوجد RESUME بدون خطأ.
24	وقت زائد بالنسبة للوحدة.
25	خطأ في الوحدة.
27	لا يوجد ورق.
39	يتوقع وجود CASE ELSE.
40	مطلوب متغير.
50	سريان زائد لـ FIELD.
51	خطأ داخلي.

الرمز	الرسالة
52	رقم ملف أو اسم ملف خطأ.
53	الملف غير موجود.
54	حالة ملف غير صحيحة.
55	الملف مفتوح بالفعل.
56	عبارة FIELD نشطة.
57	خطأ في مدخلات أو مخرجات الوحدة.
58	الملف موجود فعلاً.
59	طول سجل خطأ.
61	القرص مملوء.
62	تزيد المدخلات المطلوبة عن نهاية الملف.
63	رقم سجل غير صحيح.
64	اسم ملف خطأ.
67	ملفات كثيرة جداً.
68	الوحدة غير متاحة.
69	سريان زائد في الذاكرة الاحتياطية للاتصالات.
70	تم سحب التصريح.
71	القرص غير معد.
72	خطأ في الوسط.
73	معالم مطورة غير متاحة.
74	إعادة تسمية عبر الأقراص.
75	خطأ في الاتصال بالملف أو المسار.
76	المسار غير موجود.

وفيما يلي قائمة برسائل الخطأ التي ينتجها بيسك السريع أثناء الاستدعاء والترجمة ويتبعها وصف موجز عندما يكون ذلك مناسباً.

Advanced features unavailable

معالم مطورة غير موجودة

(لا يمكن استخدام سمات جديدة للغة مع صيغ بيسك القديمة)

Argument-count Mismatch

عداد القائمة غير متوافق

Array already dimensioned

المنظومة لها أبعاد بالفعل

Array not defined

المنظومة غير معرفة

Array not dimensioned

المنظومة لا أبعاد لها

Array too big

المنظومة كبيرة جداً

(إستخدم خيار /ah أثناء الترجمة للمنظومات التي حجمها أكبر من 64K)

Argument-count AS clause expected

عداد القائمة متوقع

AS clause required on first declaration

جزء AS مطلوب في أول توضيح

AS missing

AS غير موجود

Astrisk missing

النجمة غير موجودة

Bad file name

حالة ملف غير صحيحة

(استخدام عبارات مدخلات أو مخرجات غير مناسبة مع حالة ملف معينة)

Bad file mode

اسم ملف غير صحيح

Bad file name or number

رقم ملف أو اسم ملف غير صحيح

Bad record length

طول خطأ للسجل

Bad record number

رقم خطأ للسجل

BASE missing

BASE غير موجودة

Binary source file

ملف مصدر ثنائي

Block IF without END IF

مجموعة IF بدون END IF

Buffer size expected after /c:

حجم الذاكرة الاحتياطية متوقع بعد : /c

(لا يمكن ترجمة ملفات ثنائية. يجب أن تحفظ الملفات من بيسك المطور مع خيار A للتأكد من أن

الملف من نوع ASCII)

BYVAL allowed only with numeric argument
 مسموح بـ BYVAL مع قوائم عددية فقط

/C: buffer size too large
 حجم الذاكرة الاحتياطية كبير جداً

Cannot continue
 لا يمكن الاستمرار.

Cannot find file (file name). Input path:
 لا يمكن إيجاد مسار مدخلات الملف (اسم الملف)

(التغييرات التي حدثت في البرنامج تمنع من استمرار التنفيذ من عند هذه النقطة).

Cannot generate listing for binary BASIC Source file
 لا يمكن إنتاج قائمة ملف مصدر ثنائي بالبيسك.

Cannot start with "FN"
 لا يمكن أن تبدأ بـ FN.

CASE ELSE expected
 CASE ELSE متوقعة.

CASE without SELECT
 CASE بدون SELECT.

Choose New from Edit menu to create new SUB or function.
 اختر New من قائمة Edit لإنتاج برنامج فرعى جديد أو دالة جديدة.

Colon expected after /c
 يتوقع وجود نقطتين رأسيين بعد /c

Comma missing
 الفاصلة غير موجودة.

COMMON and DECLARE must preced executable statements.
 يجب أن تسبق COMMON و DECLARE عبارات قابلة للتنفيذ

COMMON in Quick library too small
 المشاركة في المكتبة السريعة صغيرة جداً.

COMMON name illegal
 اسم المشاركة غير صحيح.

Communication-buffer overflow.
 سريان زائد في الذاكرة الاحتياطية للاتصالات.

CONST/DIM SHARED follows SUB/FUNCTION.
 ثابت أو بعد مشترك يتبع دالة أو برنامج فرعى.

Control structure in IF.. THEN.. ELSE in complete.	تكوين التحكم في IF..THEN..ELSE غير كامل.
Data-Memory overflow.	سريان زائد لذاكرة البيانات
DECLARE required	مطلوب DECLARE
DEF FN not allowed in control statement.	غير مسموح بـ DEF FN في عبارات تحكم.
CASE و IF..THEN.. ELSE	(غير مسموح باستخدام DEF FN داخل تكوينات التحكم مثل IF..THEN.. ELSE و CASE)
DEF without END DEF	DEF بدون END DEF
DEFTYPE character specification illegal	مواصفة نوع DEF غير صحيحة
Device fault	خطأ في الوحدة (خطأ في نظم المكونات)
Device I/O error	خطأ في مدخلات أو مخرجات الوحدة
Device timeout	وقت زائد للوحدة
Device unavailable	الوحدة غير متاحة
Disk full	القرص مملوء
Disk-media error	خطأ في وسط القرص
Disk not ready	القرص غير معد
Division by zero	قسمة على صفر
DO without LOOP	DO بدون LOOP
Document too large	الوثيقة كبيرة جداً
Duplicate definition	تعريف مزدوج
Duplicate label	اسم مزدوج
Dynamic array elements illegal	عناصر منظومة ديناميكية غير مسموح بها (لا تستخدم \$VARPTR)
Element not defined	العنصر غير معرف
ELSE without IF	ELSE بدون IF

ELSEIF without IF	IF بدون ELSEIF
END DEF without DEF	IF بدون END DEF
END IF without IF	IF بدون ENF IF
END SELECT without SELECT	SELECT بدون END SELECT
END SUB or END FUNCTION must be a last line in window	يجب أن يكون END SUB أو END FUNCTION هو آخر سطر في النافذة.
END SUB/FUNCTION without SUB/FUNCTION	END SUB أو END FUNCTION بدون SUB أو FUNCTION
END TYPE without TYPE	TYPE بدون END TYPE
Equal sign missing	إشارة التساوى غير موجودة.
Error during QuickBASIC initial iza-tion	خطأ أثناء بدء بيسك السريع
Error in loading file (file)-Cannot find file	خطأ في تحميل الملف - الملف غير موجود
Error in loading file (file) - Disk I/O error	خطأ في تحميل الملف - خطأ في مدخلات أو مخرجات القرص
Error in loading file (file (file)- DOS memory-arena error	خطأ في تحميل الملف - خطأ في منطقة الذاكرة لـ DOS
Error in loading file (file) - Invalid format	خطأ في تحميل الملف - تشكيل غير صحيح.
Error in loading file (file)- Out of memory.	خطأ في تحميل الملف - الذاكرة لا تكفى.
EXIT DO not within DO.. LOOP	جزء EXIT DO غير موجود داخل دورة DO.
EXIT not within FOR.. NEXT	جزء EXIT غير موجود داخل لورة FOR
Expected: item	عنصر متوقع.
(خطأ تكويني، نقطة البداية موجودة عند العنصر)	

Expression too complex	التعبير معقد جداً
Extra file name ignored	اسم الملف الزائد أهمل
FIELD overflow	سريان زائد في FIELD
FIELD statement active	عبارة FIELD نشطة

(لا يمكن استخدام GET أو PUT مع ملف وعبارة FIELD نشطة)

File already exists	الملف موجود بالفعل
File already open	الملف مفتوح فعلاً
File not found	الملف غير موجود.
File previously loaded	سبق تحميل الملف
Fixed-length string illegal	سلسلة ثابتة الطول غير صحيحة
FOR index variable already in use	متغير دليل FOR مستخدم فعلاً
FOR index variable illegal	متغير دليل FOR غير صحيح
FOR without NEXT	FOR بدون NEXT
Formal parameter specification illegal	مواصفة مؤشر رسمي غير صحيح
Formal parameters not unique	مؤشرات رسمية ليست فريدة
Function already defined	الدالة معرفة بالفعل
Function name illegal	اسم دالة غير صحيح
Function not defined	دالة غير معرفة
GOSUB missing	GOSUB غير موجودة.
GOTO missing	GOTO غير موجودة.
GOTO or GOSUB expected	يتوقع وجود GOTO أو GOSUB
Identifier cannot end with %, &, !, #, or \$	لا يمكن للمعرف أن ينتهي بأى من % أو & أو ! أو # أو \$
Identifier cannot include period	لا يمكن أن توجد نقطة في المعرف
Identifier expected	يتوقع وجود معرف
Identifier too long	المعرف طويل جداً

Illegal function call	استدعاء خطأ لدالة
Illegal in direct mode	خطأ في الحالة المباشرة (لا تدعم النافذة الحالية ذلك)
Illegal in procedure or DEF FN	خطأ في إجراء أو في DEF FN
Illegal number	رقم غير صحيح
Illegal outside of SUB, FUNCTION, or DEF FN	خطأ خارج أو SUB أو FUNCTION DEF FN
Illegal outside of SUB/FUNCTION	خطأ خارج SUB أو FUNCTION
Illegal outside of TYPE block	خطأ خارج مجموعة TYPE
Illegal type character in numeric constant	خطأ وجود رمز من النوع الحرفي في ثابت عددي
\$INCLUDE- file access error	خطأ في الاتصال بملف شمول
Include file too large	ملف الشمول كبير جداً
Input file not found	ملف المدخلات غير موجود
INPUT missing	INPUT مفقودة
Input past end file	مدخلات تزيد عن نهاية الملف
Input run-time module path: BRUN 40 is not found	مسار جزء وقت تنفيذ المدخلات (BRUN40). EXE غير موجود
Integer between 1 and 32,767 required	مطلوب رقم صحيح يقع بين 1 و 32,767
internal error	خطأ داخلي
Internal error near XXX	خطأ داخلي بالقرب من XXX
Invalid character	رمز غير صحيح
Invalid constant	ثابت غير صحيح
Invalid DECLARE for BASIC procedure	توضيح غير صحيح لإجراء بيسك

(لا يمكن استخدام BYVAL أو ALIAS أو CDECL مع إجراءات بيسك)

Label not defined	اسم غير معرف
Label not defined: label	اسم غير معرف: إسم
Left parantheses missing	الأقواس اليسرى غير موجودة.
Line invalid. Start again	سطر غير صحيح، ابدأ مرة أخرى
Line number or label missing	رقم سطر أو اسم سطر غير موجود
Line too long	السطر طويل جداً
LOOP without DO	LOOP بدون DO
Lower bound exceeds upper bound	الحد السفلى يتعدى الحد العلوى
Math overflow	سريان زائد حسابى
\$Metacommand error	خطأ شبيه الأمر
Minus sign missing	إشارة سالبة غير موجودة.
Missing Event Trapping (/W) or checking Between Statements (/V) option	خيار اصطياد الأحداث (/W) أو التأكد بين العبارات (/V) مفقود
Missing an error (/E) option	خيار عند حدوث خطأ (/E) غير موجود.
Missing resume next (/X) option	خيار افترض التالى (/X) غير موجود
Missing level code too large	رمز المستوى طويل جداً غير موجود
Module not found. Unload module from program?	الجزء غير موجود، هل يراد عدم تحميل جزء من البرنامج
Must be first statement on the line	يجب أن تكون أول عبارة على السطر
Name of subprogram illegal	اسم البرنامج الفرعى غير صحيح
Nested function definition	تعريف دالة متداخلة
NEXT missing for variable	NEXT غير موجودة لمتغير
NEXT without FOR	NEXT بدون FOR
No line number in module-name at address segment: offset	لا يوجد رقم سطر فى اسم الجزء عند عنوان القطاع : فرع

No main module. Choose Set Main Module from the Run menu to select one	لا يوجد جزء رئيسي، اختر أعداد جزء رئيسي من قائمة Run لاختيار واحد
No RESUME	لا يوجد RESUME
Not watchable	غير مرئي
(المتغير الذي تم اختياره لتنفيذ الرؤية غير مناسب)	
Numeric array illegal	منظومة عددية غير صحيحة
Only simple variables allowed	مسموح بمتغيرات بسيطة فقط
Operation requires disk	العملية تتطلب قرصاً
Option unknown: Option	الخيار غير معروف: خيار
Out of DATA	لا توجد عبارة DATA
Out of data space	لا يوجد مكان للبيانات
Out of memory	لا توجد ذاكرة كافية
Out of paper	لا يوجد ورق
Out of stack space	مكان الحزمة أصبح غير كافياً
Out of string space	مكان السلسلة أصبح غير كافياً
Overflow	سريان زائد
Overflow in numeric constant	سريان زائد في ثابت عددي
Parameter type mismatch	نوع غير متوافق للمؤشر
Path not found	المسار غير موجود
Path/ file access error	خطأ اتصال بملف أو مسار
Permission denied	التصريح تم الغاؤه
Procedure already defined in Quick library	الإجراء معرف فعلاً في المكتبة السريعة
Procedure too large	الإجراء كبير جداً
Program-memory overflow	سريان زائد لذاكرة البرنامج
Read error on standard input	خطأ قراءة في مدخلات نمطية

Record/ string assignement required

مطلوب تحديد سجل أو سلسلة

Redo from start

ابدأ من البداية

(أعطيت استجابة غير صحيحة لعبارة INPUT)

Rename access disks

إعادة تسمية عبر الأقراص

(لا يمكن إعادة تسمية ملف مع تحديد مشغل جديد)

Require DOS 2.0 or later

يتطلب DOS صيغة 2.0 أو أعلى

(صيغة غير صحيحة لنظام التشغيل DOS مستخدمة للعمل في هذه الحالة)

RESUME without error

RESUME بدون خطأ

RETURN without GOSUB

RETURN بدون GOSUB

Right parantheses missing

الأقواس اليمنى غير موجودة

SEG or BYVAL not allowed in

SEG و BYVAL غير مسموح بها في

CALLS

CALLS

SELECT without END SELECT

SELECT بدون END SELECT

Semicolon missing

فاصلة منقوطة غير موجودة

Separator illegal

فاصل غير صحيح

Simple or array variable expected

يتوقع متغير بسيط أو منظومة

Skipping forward to END TYPE

التقدم للأمام حتى عبارة END TYPE

statement

(خطأ في عبارة TYPE و END TYPE يجعل بيسك السريع يهمل بقية التعريف)

Statement cannot occur within

العبارة لا يمكن أن تحدث داخل ملف شمول

\$INCLUDE file

Statement cannot precede SUB/

العبارة لا يمكن أن تسبق تعريف البرنامج

FUNCTION definition

الفرعى أو الدالة

Statement ignored

تهمل العبارة

Statement illegal in TYPE block	عبارة غير صحيحة فى مجموعة TYPE
Statement unrecognized	عبارة غير مميزة
Statements/ labels illegal between SELECT CASE and CASE	عبارات أو أسماء غير صحيحة بين SELECT CASE و CASE.
STOP in module name at address segment/ offset	توقف فى اسم الجزء عند عنوان القطاع / الفرع
String assignment required	مطلوب تحديد سلسلة
String constant required for ALIAS	مطلوب ثابت سلسلة لـ ALIAS
String expression required	مطلوب تعبير سلسلة
String formula too complex	صيغة سلسلة معقدة جداً
String space corrupt	مكان السلسلة تلف
String variable required	مطلوب متغير سلسلة
SUB or FUNCTION missing	SUB أو FUNCTION غير موجودة
SUB/ FUNCTION without END SUB/ FUNCTION	برنامج فرعى أو دالة بدون END SUB أو END FUNCTION
Subprogram error	خطأ برنامج فرعى
Subprogram not defined	برنامج فرعى غير معرف
Subprograms not allowed in control statements	غير مسموح ببرامج فرعية فى عبارات التحكم
Subscript out of range	دليل المنظومة يقع خارج المدى المحدد له
Subscript syntax illegal	تكوين الدليل غير صحيح
Syntax error	خطأ تكوينى
Syntax error in numeric constant	خطأ تكوينى فى ثابت عددى
THEN missing	THEN غير موجودة
TO missing	TO غير موجودة
Too many arguments in function call	قيم كثيرة جداً فى نداء الدالة

Too many dimensions	أبعاد كثيرة جداً
Too many files	ملفات كثيرة جداً
Too many labels	أسماء كثيرة جداً
Too many named COMMON blocks	مجموعات مشاركة مسماة كثيرة جداً
Too many TYPE definitions	تعريفات نوع كثيرة جداً
Too many variables for INPUT	متغيرات كثيرة جداً لعبارة المدخلات
Too many variables for LINE INPUT	متغيرات كثيرة جداً لعبارة مدخلات السطر
Type mismatch	النوع غير متوافق
TYPE missing	TYPE غير موجودة
Type more than 65,535 bytes	مكتوب أكثر من 65,535 بايت
Type not defined	النوع غير معرف
TYPE statement improperly nested	عبارة TYPE متداخلة بصورة غير مناسبة
TYPE without END TYPE	TYPE بدون END TYPE
Typed variable not allowed in expression	المتغير المكتوب غير مسموح به في التعبير
Unexpected end of file in TYPE declaration	نهاية غير متوقعة الملف في توضيح النوع
Unprintable error	خطأ غير قابل للطباعة
Unrecognized switch error: "QU"	خطأ مفتاح غير مميز "Qu"
Valid options: [RUN] file /AH/B/C: Buf/G/H/L [lib]/mbf/cmd string	سلسلة خيارات صحيحة
Variable-length string required	مطلوب سلسلة متغيرة الطول
Variable name not unique	اسم متغير غير فريد
Variable required	مطلوب متغير

(متغير غير موجودة في عبارة INPUT أو LET أو READ أو SHARED أو GET و PUT)

WEND without While

WHILE without WEND

Wrong number of dimensions

WHILE بدون WEND

WEND بدون WHILE

عدد خطأ من الأبعاد

ملحق G

تقارن على بيسك السريع

١ - حول هذا الكتاب

- أ - إلى أى شخص موجه هذا الكتاب؟
- ب - إذا كنت تتعلم بيسك السريع فما هو التسلسل الذى يجب أن تتبعه فى دراسة هذا الكتاب؟
- ج - ما هى دلالة الخطوات المرقمة فى قسم العمليات التقليدية فى كل درس؟

٢ - عرض عام لبيسك السريع

- أ - ما هى السمات الخاصة ببيسك السريع - الصيغة 4.0؟
- ب - ما هى التوسعات فى البيسك الموجودة فى بيسك السريع - الصيغة 4.0؟
- ج - ما هى الطرق الأخرى الممكنة لوضع بيسك السريع على القرص الصلب؟

٣ - عينة لجلسة مع بيسك السريع

- أ - ماذا تكتب لبدء بيسك السريع من ملقن DOS؟
- ب - كيف تكتب برنامجاً بلغة بيسك السريع؟
- ج - ما هى علامات التنقيط التى تفصل أسطر برنامج بيسك السريع؟ ومتى تكون على نفس المستوى الطبيعى؟
- د - كيف تنفذ برنامجاً فى بيسك السريع؟
- هـ - ما هى القوائم التى تستدعى عندما تسحب لأسفل من الاختيار؟ وكيف يمكنك أن تحفظ ملفاً على مشغل أقراص مختلف أو فى دليل مختلف؟

٤ - الترجمة من DOS

- أ - تحت أى ظروف تتم ترجمة البرنامج من DOS؟
- ب - ما هو الغرض من التوصيل Linking؟
- ج - ما هو نوع الملف الذى ينتج من هذا النوع من الترجمة؟

٥ - دالة ABS

- أ - ما هو نوع القيمة التي تعيدها دالة ABS؟
- ب - ما هي المواقع الأخرى التي تهمل إشارة العدد؟

٦ - دالة ASC

- أ - ما هو مدى القيمة التي تعيدها دالة ASC؟
- ب - ما هو متمم دالة ASC؟
- ج - ما هي الرموز غير المتوقعة؟

٧ - دالة ATAN

- أ - ما هي وحدة القيمة التي تعيدها دالة ATAN؟
- ب - ما هو النوع التقليدي للقيمة التي تعيدها الدالة؟

٨ - عبارة BEEP

- أ - كم عدد مرات إنتاج عبارة BEEP صوتاً من الكمبيوتر؟
- ب - ما هو نوع المواقع التي يمكن استخدام عبارة BEEP فيها؟

٩ - عبارتا BLOAD و BSAVE

- أ - ما هو جزء الفرع offset في تكوين عبارة BLOAD؟
- ب - ما هو جزء مدى الطول range of length في تكوين عبارة BSAVE؟
- ج - ما هو نوع البيانات التي تتعامل معها عبارة BLOAD وعبارة BSAVE؟
- د - ما هي الوحدات غير المدعومة لعبارتي BLOAD و BSAVE؟

١٠ - عبارتا CALL و CALLS

- أ - ما هو أقصى طول لاسم البرنامج الفرعي؟
- ب - ما هو الفرق بين استخدام BYVAL و SEG في عبارة CALL؟
- ج - ما هي إجراءات اللغة التي يمكن استدعاؤها باستخدام عبارات CALL و CALLS؟

د - متى لا تستخدم الكلمة المحجوزة CALL فى عبارة CALL؟
هـ - ما هى البرمجة للغات مختلطة؟

١١ - عبارة CALL ABSOLUTE

أ - إلى أين ينتقل التحكم فى البرنامج عندما تنفذ عبارة CALL ABSOLUTE؟
ب - ما هى المكتبة السريعة التى يجب أن تقوم بتحميلها لكى تعمل هذه العبارة؟
ج - كيف تمرر القوائم إلى الجزء المندى عليه؟

١٢ - عبارتا CALL INT86OLD و CALL INTERRUPT

أ - ما هو استدعاء عمل DOS؟
ب - ما هو الازعاج interrupt؟
ج - ما هو الفرق بين عناصر المنظومة المستخدمة فى INT86OLD و INT86XOLD؟

د - ما هى المكتبة السريعة التى يجب أن تقوم بتحميلها لكى تعمل هذه العبارات؟
هـ - فى أى الأغراض تستخدم هذه العبارات؟

١٣ - دوال CDBL و CINT و CLNG و CSNG

أ - إلى أى نوع تحول دالة CDBL القائمة؟
ب - إلى أى نوع تحول دالة CINT القائمة؟
ج - إلى أى نوع تحول دالة CLNG القائمة؟
د - إلى أى نوع تحول دالة CSNG القائمة؟

١٤ - عبارة CHAIN

أ - هل يمكن لبرنامج (.EXE) مستقل بذاته أن يستدعى ملفات مصدر مكتوبة ببيسك السريع؟
ب - هل يستطيع برنامج ينفذ فى بيئة ببيسك السريع أن يستدعى برنامجاً قائماً بذاته؟
ج - كيف تقتسم المتغيرات مع برنامج متسلسل مع أجزاء أخرى؟
د - هل يعود تحكم البرنامج إلى برنامج متسلسل مع أجزاء أخرى؟

١٥ - دالة CHR\$

- أ - ما هو نوع القيمة التي تعيدها دالة CHR\$؟
- ب - ما هو مدى القيم للقائمة؟
- ج - ما هي النافذة الفورية؟

١٦ - عبارة CIRCLE

- أ - ما هو جزء aspect في التكوين؟
- ب - ما هي الأجزاء التي تقرر إذا ما كانت الإحداثيات مطلقة أم نسبية؟
- ج - كيف يمكنك أن ترسم قوساً مستخدماً عبارة CIRCLE؟

١٧ - عبارة CLEAR

- أ - ماذا يحدث عندما تنفذ عبارة CLEAR بدون أى قائمة؟
- ب - ما هي الرصة؟
- ج - كيف يمكنك أن تغير من حجم الرصة باستخدام عبارة CLEAR؟
- د - متى تريد أن تغير من حجم الرصة؟

١٨ - عبارة CLOSE

- أ - ماذا يحدث عندما تستخدم عبارة CLOSE بدون قائمة؟
- ب - ما هي العبارات الأخرى التي تغلق الملفات؟
- ج - هل يمكن استخدام رقم الملف بعد أن يغلق الملف المناظر لهذا الرقم؟
- د - هل الذاكرة الاحتياطية النهائية تكتب على القرص قبل اغلاق الملف؟

١٩ - عبارة CLS

- أ - أين يكون موقع نقطة البداية بعد تنفيذ عبارة CLS؟
- ب - ما هي القيم التي يمكن تمريرها إلى عبارة CLS؟

٢٠ - عبارة COLOR

- أ - ما هو الغرض من عبارة COLOR؟
- ب - ما هي حالات الشاشة التي تدعم حدود الشاشة؟
- ج - ما هو نوع الألوان التي لا يمكن استخدامها كخلفية؟

٢١ - دالة COMMAND\$

- أ - ما هي الحالة التي تعود بها دالة COMMAND\$ لمؤشر سطر الأوامر؟
- ب - ما معنى القراءة من عناصر مميزة Parsing؟
- ج - متى تستخدم قوائم سطر الأوامر؟

٢٢ - عبارة COMMON

- أ - ما هو الفرق بين مجموعة COMMON مسماة وأخرى غير مسماة؟
- ب - متى تستخدم مجموعة COMMON المسماة؟
- ج - متى تستخدم مجموعة COMMON غير المسماة؟
- د - من الذى يسبق الآخر فى مجموعة COMMON، اسماء المعارف أو ترتيب قائمة المعارف؟

٢٣ - الثابت CONST

- أ - ما هي الأنواع المختلفة للثوابت؟
- ب - ما هي أنواع البيانات المختلفة التى يمكن أن تكون ثابتاً؟
- ج - متى تستخدم الثوابت؟
- د - متى تستخدم ثابتاً حرفياً بدلاً من ثابت رمزى؟

٢٤ - دالة COS

- أ - ما هو النوع التقليدى للنتيجة من دالة COS؟
- ب - ما هي وحدة القيمة التى تعود من دالة COS؟
- ج - ما هو نوع دالة COS؟

٢٥ - دالة CSRLIN

- أ - ما هي المعلومات التى تعيدها دالة CSRLIN؟
- ب - فى أى غرض يمكنك أن تستخدم دالة CSRLIN؟
- ج - فى البرنامج المستخدم فى قسم العملية التقليدية ما هي المؤشرات المستخدمة فى FUNCTION GCh\$

٢٦ - دوال CVD و CVI و CVL و CVS

- أ - ما هو نوع القيمة التي تعيدها الدالة CVD؟
- ب - ما هو نوع القيمة التي تعيدها الدالة CVI؟
- ج - ما هو نوع القيمة التي تعيدها الدالة CVL؟
- د - ما هو نوع القيمة التي تعيدها الدالة CVS؟
- هـ - ما هي الدوال التي تكون عكس هذه الدوال؟
- و - ما هي الطريقة الأسهل لتخزين واسترجاع بيانات باستخدام ملف على قرص؟

٢٧ - عبارة DATA

- أ - ما هو عدد عناصر البيانات التي يجب أن يكون في عبارة DATA لتجنب خطأ وقت التنفيذ؟
- ب - مع أى عبارة تستخدم عبارة DATA؟
- ج - ماذا يحدث إذا ما احتوت عبارة DATA على عناصر أكثر من اللازم لعبارة READ؟

٢٨ - دالة DATE \$

- أ - ما هو شكل قائمة عبارة DATE\$؟
- ب - في أى شيء تستخدم دالة DATE\$؟
- ج - ما هو شكل القيمة التي تعيدها دالة DATE\$؟

٢٩ - عبارة DECLARE

- أ - ما هو الغرض من عبارة DECLARE؟
- ب - ما نوع السلاسل غير المسموح بها في قائمة مؤشرات عبارة DECLARE؟
- ج - كيف تسرد متغيرات المنظومات في قائمة المؤشرات؟
- د - ماذا يحدث إذا لم تكن عبارة DECLARE مكتوبة من قبل المبرمج؟
- هـ - ماذا يحدد جزء CDCEL؟
- و - في أى شيء يستخدم جزء ALIAS؟
- ز - ما هو DGROU؟

- ح - ماذا يمكن أن يتسبب في نقل متغير في الذاكرة؟
ث - ما هي الطريقة الجيدة لتجميع عبارات DECLARE؟

٣٠ - عبارة DEF FN

- أ - كيف تستدعي الدالة؟ متى تعرف بعبارة DEF FN؟
ب - كيف يمكنك أن تحدد نوع نتيجة الدالة؟
ج - ما هو الفرق بين تمرير مؤشرات بواسطة دليل أو بواسطة قيمة؟
د - كيف يمكنك أن تترك دالة معرفة بواسطة DEF FN بصورة نهائية؟

٣١ - DEF SEG

- أ - كيف يحسب عنوان الذاكرة؟
ب - ما هو جزء address في تكوين DEF SEG؟
ج - ما هو جزء مدى العنوان؟
د - ما هو القطاع التقليدي؟

٣٢ - عبارات DEFDBL ، DEFINT ، DEFLNG ، DEFSNG ، DEFSTR

- أ - من أي نوع توضح عبارة DEFDBL مجموعة من الرموز؟
ب - من أي نوع توضح عبارة DEFINT مجموعة من الرموز؟
ج - من أي نوع توضح عبارة DEFLNG مجموعة من الرموز؟
د - من أي نوع توضح عبارة DEFSNG مجموعة من الرموز؟
هـ - من أي نوع توضح عبارة DEFSTR مجموعة من الرموز؟
و - ما هي دلالة حالة الحروف في هذه العبارات؟
ز - كيف يمكنك أن تغير نوع الحرف بعد أن تقوم بتعريفه فعلاً؟

٣٣ - عبارة DIM

- أ - في أي شيء تستخدم عبارة DIM؟
ب - ما هو أقصى عدد مسموح به لأبعاد إحدى المنظومات؟

- ج - ما هو أقصى حجم لمنظومة معتادة؟
- د - كيف توضح منظومة يكون حجمها أكبر من أقصى حجم؟
- هـ - ما هي الأنواع المختلفة للمنظومات؟

٣٤ - عبارة DO LOOP

- أ - ما هي الدورة؟
- ب - ما هو تكوين التحكم؟
- ج - ما هو تعبير بوليان؟
- د - ما هو الاختلاف في منطق برنامج يستخدم عبارة دورة DO تتبع التكوين الأول عن دورة DO تتبع التكوين الثاني؟

٣٥ - عبارة DRAW

- أ - ما هي وحدة الحركة التقليدية في عبارة DRAW؟
- ب - كيف يمكنك استخدام "TA" في ماكرو DRAW؟
- ج - ما هو تنفيذ سلسلة جزئية؟
- د - كيف يمكنك أن تغير وحدة الحركة في عبارة DRAW؟
- هـ - كيف يمكنك وصف ما إذا كانت الاحداثيات مطلقة أو نسبية؟

٣٦ - عبارة END

- أ - ما هي الاستخدامات المختلفة لعبارة END؟
- ب - متى يمكن حذف عبارة END دون أن يحدث خطأ وقت التنفيذ؟

٣٧ - عبارة ENVIRON\$ ودالة ENVIRON

- أ - ما هي القيمة التي تعيدها دالة ENVIRON؟
- ب - كيف توضح متغيرات بيئة DOS من DOS؟
- ج - كيف توضح متغيرات بيئة DOS من برنامج بيسك السريع؟
- د - ماذا يحدث لاعدادات البيئة بعد انتهاء برنامج بيسك السريع؟
- هـ - كيف تحصل على اعداد ملقن DOS الحالي من برنامج بيسك السريع؟

٣٨ - دالة EOF

- أ - ماذا يحدث إذا حاولت أن تقرأ ملفاً بعد استنفاد البيانات الموجودة في الملف؟
- ب - ماذا يؤدي إلى حدوث شرط نهاية الملف على وحدة اتصالات عندما تفتح الوحدة كملف ASCII؟
- ج - ما هي الوحدات التي لا يمكنك استخدامها مع دالة EOF؟

٣٩ - عبارة ERASE

- أ - في أي شيء تستخدم عبارة ERASE؟
- ب - ما هو نوع المنظومات التي يمكنك استخدام عبارة ERASE معها؟
- ج - ماذا يحدث للمنظومات الديناميكية عندما تنفذ ERASE عليها؟
- د - ماذا يحدث للمنظومات الاستاتيكية عندما تنفذ ERASE عليها؟

٤٠ - دالتا ERDEV و ERDEV\$

- أ - على أي نوع من الوحدات تستخدم دوال ERDEV و ERDEV\$؟
- ب - كيف يمكنك تفسير القيمة التي تعود من دالة ERDEV؟

٤١ - دالتا ERR و ERL و عبارة ON ERROR

- أ - ما هو الانتخاب polling؟
- ب - ما هي المعلومات التي تعيدها دالة ERR؟
- ج - ما هي المعلومات التي تعيدها دالة ERL؟
- د - كيف يمكنك عمل تصيد للأحداث خاص بالأخطاء؟

٤٢ - عبارة ERROR

- أ - لماذا يجب عليك أن تعرف شفرة أخطائك الخاصة بك؟
- ب - ما هي الطريقة الآمنة لتحديد شفرات الأخطاء؟
- ج - كيف يمكنك استخدام عبارة ERROR في تعريف شفرات خطأ جديدة؟

٤٣ - عبارة EXIT

- أ - لماذا يجب أن تخرج من إحدى الدورات بصفة دائمة؟
- ب - ما هي الأشكال المختلفة لعبارة EXIT؟

٤٤ - دالة EXP

- أ - فى أى شىء تستخدم دالة EXP؟
- ب - ما هو مدى القائمة فى دالة EXP؟

٤٥ - عبارة FIELD

- أ - ماذا تفعل عبارة FIELD؟
- ب - ما هى الطريقة السهلة لتحقيق نفس تأثير عبارة FIELD؟
- ج - لماذا تكون عبارة FIELD أكثر نفعاً مع الملفات غير النصية؟

٤٦ - دالة FILEATTR

- أ - ما هى معالجة ملف DOS؟
- ب - ما هى الخواص التى تعود من معنى دالة FILEATTR؟
- ج - أين يمكنك أن تستخدم دالة FILEATTR؟

٤٧ - عبارات FILES و CHDIR و MKDIR و RMDIR

- أ - ماذا تفعل عبارة FILES؟
- ب - كيف تمرر سلاسل مواصفات الملف إلى العبارات فى هذا الجزء؟
- ج - ما هى الرموز الخاصة wild-card؟
- د - ما هو أقصى طول لاسم المسار فى عبارة CHDIR؟
- هـ - ما هو أقصى طول لاسم المسار فى عبارة RMDIR؟

٤٨ - دالة FIX

- أ - ما هو نوع النتيجة التى تعيدها دالة FIX؟
- ب - ما هى التطبيقات التى تتطلب دالة FIX؟

٤٩ - عبارة FOR.. NEXT

- أ - متى يستخدم جزء STEP؟
- ب - هل تستطيع أن تستخدم أرقاماً كسرية كقيم للعداد؟
- ج - ما هو التداخل؟

٥٠ - دالة FRE

- أ - ما هي المعلومات التي تقدمها دالة FRE؟
- ب - كيف تجد حجم المكان المتاح للرمز؟
- ج - تحت أي ظروف تستخدم دالة FRE؟
- د - كيف تجعل كل مخازن السلاسل متماسة باستخدام دالة FRE؟

٥١ - دالة FREEFILE

- أ - في أي غرض تستخدم دالة FREEFILE؟
- ب - ما هي المعلومات التي تقدمها دالة FREEFILE؟

٥٢ - عبارة FUNCTION

- أ - ما هو أقصى طول مسموح به لاسم الدالة؟
- ب - ماذا يفعل جزء STATIC؟
- ج - كيف تسرد مؤشرات المنظومة؟
- د - كيف تترك مجموعة FUNCTION للتنفيذ؟
- هـ - متى تكون الدالة متسمة بسمة الاعادة الذاتية؟

٥٣ - عبارات GET و PUT

- أ - متى تستخدم عبارات GET و PUT في تشغيل الملف؟
- ب - ما هو جزء recordnum في تكوين عبارة GET؟
- ج - لماذا يجب عدم استخدام عبارة FIELD عند استخدام عبارة GET مع جزء variable؟
- د - لماذا يجب أن تتجنب استخدام عبارات GET و PUT في الاتصالات مع سجلات ثابتة الطول؟

٥٤ - عبارات GET graphics و PUT graphics

- أ - ما هو الغرض من جزء STEP في تكوين عبارات GET و PUT؟
- ب - ما هي أهمية جزء array في تكوين عبارة PUT؟
- ج - ما هي مؤثرات بوليان التي يمكن استخدامها مع عبارة PUT؟

٥٥ - تكوين GOSUB.. RETURN

- أ - ما هو نوع البرامج الفرعية التي يمكنك استدعاؤها باستخدام عبارة GOSUB؟
- ب - ما مدى العمق الذي يمكن استخدامه في تداخل البرامج الفرعية؟
- ج - لماذا يجب أن تكتب برامج فرعية؟
- د - كيف تستخدم عبارة RETURN مع اسم سطر؟

٥٦ - عبارة GOTO

- أ - ما هو التفرع غير الشرطي؟
- ب - متى يمكنك استخدام التفرع غير الشرطي؟
- ج - كيف يمكنك العودة إلى نفس المكان بعد تنفيذ عبارة GOTO؟

٥٧ - دالة HEX\$

- أ - هل تستطيع أداء حسابات ستة عشرية على قيم تعيدها دالة HEX\$؟
- ب - ما هو نوع القائمة التي تقبلها دالة HEX\$؟

٥٨ - عبارة IF.. THEN.. ELSE

- أ - ما هو التفرع الشرطي؟
- ب - ما هو الفرق بين تكويني عبارة IF.. THEN.. ELSE؟
- ج - هل تستطيع أن تستخدم GOSUB مع THEN أثناء الإشارة إلى اسم السطر؟

٥٩ - عبارة \$INCLUDE

- أ - ماذا يعنى شبيه الأمر؟
- ب - ما هي القيود على الملف المشمول؟
- ج - هل تستطيع تقديم اسم مسار مع اسم ملف في عبارة \$INCLUDE؟

٦٠ - دالة INKEY\$

- أ - ما هي المعلومات التي تعيدها دالة INKEY\$؟
- ب - كيف تحذف خليط مشاوير المفاتيح باستخدام دالة INKEY\$؟
- ج - ما هي مشاوير المفاتيح التي لا تستطيع اصطياها؟

٦١ - عبارات INP و OUT و WAIT

- أ - بماذا تتعامل هذه الكلمات المحجوزة؟
- ب - متى تستخدم هذه الكلمات المحجوزة؟
- ج - فى عبارة WAIT فى أى ترتيب تنفذ العمليات؟

٦٢ - عبارة INPUT

- أ - كيف تسرد المتغيرات فى عبارة INPUT؟
- ب - كيف يمكنك أن تبطل طباعة علامة استفهام؟
- ج - كيف يجب إدخال المدخلات؟
- د - أثناء الإدخال ماذا يحذف محتوى المدخلات بغض النظر عن موقع نقطة البداية؟

٦٣ - عبارة INPUT#

- أ - متى تستخدم عبارة INPUT#؟
- ب - كيف تسرد المتغيرات فى هذه العبارة؟
- ج - متى تنتهى المدخلات من الملف؟

٦٤ - عبارة INPUT\$

- أ - كيف تختلف عبارة INPUT\$ عن عبارة INPUT#؟
- ب - هل تعرض عبارة INPUT\$ بيانات على الشاشة؟
- ج - ما هى الوحدات التى تدعمها عبارة INPUT\$؟

٦٥ - دالة INSTR

- أ - ما هى المعلومات التى تعيدها دالة INSTR؟
- ب - كيف تختار الموقع فى السلسلة التى يجب أن يبدأ فيه البحث؟

٦٦ - دالة INT

- أ - ما هو نوع النتيجة التى تعيدها دالة INT؟
- ب - هل تكون للنتيجة اشارة؟

٦٧ - دالتا IOCTL\$ و IOCTL\$

- أ - متى يمكن استخدام هاتين الدالتين؟
- ب - ما هي متطلبات هاتين الدالتين لتعمل؟
- ج - ما هي الوحدات غير المدعومة بواسطة هاتين الدالتين؟

٦٨ - عبارات KEY

- أ - في أي غرض تستخدم عبارات KEY؟
- ب - كيف يمكنك تعريف مشوار مفتاح جديد باستخدام عبارات KEY؟
- ج - كيف يمكنك إلغاء عمل تحديد سبق حدوثه بواسطة دالة KEY؟

٦٩ - عبارة KILL

- أ - هل تستطيع أن تحذف ملف يكون مفتوحاً؟
- ب - متى تستخدم الرموز الخاصة wild-card في اسم الملف؟

٧٠ - الأسماء أو LABELS

- أ - ما هي الأسماء أو العناوين labels؟
- ب - ما هو أقصى حجم مسموح به للاسم؟
- ج - ما هو عدد الأسماء التي يمكنك أن تكتبها على سطر واحد؟
- د - متى تستخدم الأسماء؟

٧١ - دالتا LBOUND و UBOUND

- أ - ما هي المعلومات التي تقدمها دالة LBOUND؟
- ب - ما هي المعلومات التي تقدمها دالة UBOUND؟
- ج - متى يمكنك استخدام هذه الدوال؟

٧٢ - دالتا UCASE\$ و LCASE\$

- أ - ما هي الاجراءات التي تنفذها هذه الدوال على السلاسل؟
- ب - متى تستخدم هذه الدوال؟

٧٣ - دالة LEFT\$

- أ - ماذا يحدث إذا كان جزء تعبير السلسلة فارغاً؟
- ب - على أى جزء من السلسلة تعمل دالة LEFT\$؟

٧٤ - دالة LEN

- أ - فى أى غرض تستخدم دالة LEN؟
- ب - أى تكوين تستخدمه للحصول على حجم متغير؟

٧٥ - عبارة LET

- أ - ما هى الوظيفة التى تؤديها عبارة LET؟
- ب - كيف يمكنك أن توضح متغيراً بدون توضيح رسمى؟
- ج - هل تستطيع أن تكتب برنامجاً بدون استخدام عبارة LET؟

٧٦ - عبارة LINE

- أ - فى عبارة LINE ما هى وظيفة جزء STEP؟
- ب - كيف يمكنك انتاج صندوق مملوء باستخدام عبارة LINE؟
- ج - كيف تعرف الاحداثيات المقدمة صندوقاً؟

٧٧ - عبارات LINE INPUT و LINE INPUT#

- أ - ما هو الاختلاف بين عبارتى LINE INPUT و LINE INPUT#؟
- ب - متى تنتهى المدخلات؟

٧٨ - دالة LOC

- أ - ما هى المعلومات التى تقدمها دالة LOC؟
- ب - على أى الوحدات لا يمكنك استخدام دالة LOC؟

٧٩ - عبارة LOCATE

- أ - فى أى غرض تستخدم عبارة LOCATE؟
- ب - كيف يمكنك التحكم فى حجم نقطة البداية؟
- ج - كيف يمكنك التحكم فى أن تكون نقطة البداية مرئية أو غير مرئية؟

٨٠ - عبارتا LOCK و UNLOCK

- أ - تحت أى ظروف يمكنك أن تستخدم عبارتى LOCK و UNLOCK؟
- ب - ما هو أقصى رقم سجل؟
- ج - ما هو أقصى حجم سجل؟

٨١ - دالة EOF

- أ - ما هى المعلومات التى تقدمها دالة EOF؟
- ب - على أى الوحدات لا يمكنك استخدام دالة EOF؟
- ج - متى يمكنك استخدام دالة EOF؟

٨٢ - دالة LOG

- أ - ما هو الأساس الذى تعيد دالة LOG نتيجتها عليه؟
- ب - ما هو النوع التقليدى للنتيجة؟

٨٣ - دالة LPOS

- أ - ما هى المعلومات التى تقدمها دالة LPOS؟
- ب - هل القيمة التى تعيدها LPOS هى الموقع الطبيعى لرأس الطابع؟

٨٤ - عبارتا LPRINT و LPRINT USING

- أ - أين تتجه المخرجات عند استخدام هذه العبارات؟
- ب - ما هو العرض الذى تفترضه هذه العبارات للطابع؟

٨٥ - عبارتا LSET و RSET

- أ - فى أى غرض تستخدم عبارتا LSET و RSET؟
- ب - ماذا يحدث عندما يكون متغير السلسلة أكبر من البيانات؟
- ج - ما هو الاختلاف بين عبارة LSET وعبارة RSET؟

٨٦ - دالة LTRIM\$ و RTRIM\$

- أ - فى أى غرض تستخدم هاتان العبارتان؟
- ب - ما هو الاختلاف بين دالة LTRIM\$ و RTRIM\$؟

٨٧ - دالة MID\$ وعبارة

- أ - ما هو الاختلاف عندما تستخدم MID\$ فى الطرف الأيسر للتحديد وعندما تستخدمها فى الطرف الأيمن من التحديد؟
- ب - كيف يمكنك أن تتحكم فى الموقع الذى تستخلص منه السلسلة الجزئية؟
- ج - كيف يمكنك أن تتحكم فى الموقع الذى تضاف فيه السلسلة الجزئية؟

٨٨ - دوال MKD\$ و MKI\$ و MKL\$ و MKS\$

- أ - ما هو نوع القيمة التى تعيدها دالة MKD\$؟
- ب - ما هو نوع القيمة التى تعيدها دالة MKI\$؟
- ج - ما هو نوع القيمة التى تعيدها دالة MKL\$؟
- د - ما هو نوع القيمة التى تعيدها دالة MKS\$؟
- هـ - مع أى عبارات أخرى تستخدم هذه العبارات؟

٨٩ - دوال MKDMBF\$ و MKSMBF\$ و CVDMBF\$ و CVSMBF\$

- أ - ماذا يعنى تشكيل ميكروسوفت الثنائى؟
- ب - ما هو نوع القيمة التى تعيدها MKDMBF\$؟
- ج - ما هو نوع القيمة التى تعيدها MKSMBF\$؟
- د - ما هو نوع القيمة التى تعيدها CVDMBF\$؟
- هـ - ما هو نوع القيمة التى تعيدها CVSMBF\$؟
- و - ما هى مميزات تشكيل MBF على تشكيل IEEE؟

٩٠ - عبارة NAME.. AS

- أ - ما هو الفرق بين أمر RENAME من DOS وعبارة NAME.. AS؟
- ب - هل يمكنك إعادة تسمية دلائل باستخدام هذه العبارة؟

٩١ - دالة OCT\$

- أ - ما هي القيمة التي تعيدها دالة OCT\$؟
- ب - هل تستطيع استخدام نتيجة دالة OCT\$ في حسابات رياضية؟

٩٢ - عبارة ON event GOSUB

- أ - ماذا يعنى اصطلياد الأحداث؟
- ب - متى تختار عملية تصيد الأحداث؟
- ج - ماذا يحدث لآلية اصطلياد الأخطاء داخل جزء معالجة الأحداث؟

٩٣ - عبارات ON.. GOTO و ON.. GOSUB

- أ - ماذا يحدد كيفية التفريع إلى أرقام أسطر أو أسماء أسطر؟
- ب - هل ترتيب أرقام الأسطر أو أسمائها معنوى؟
- ج - هل يمكنك استخدام خليط من أرقام أسطر وأسماء أسطر في عبارة ON.. GO-
SUB /GOTO ؟
- د - كيف تتشابه هذه العبارة مع عبارة IF.. THEN.. ELSE ؟

٩٤ - عبارة OPEN

- أ - ما هو الغرض من عبارة OPEN ؟
- ب - في حالة الاتصال العشوائى RANDOM ما هو ترتيب الحالات التي تحاول
عبارة OPEN أن تفتح فيها الملف قبل أن تستلم؟
- ج - ما هو القيد على حجم السجل؟
- د - ما هي الوحدات المدعمة كجزء من اسم الملف؟

٩٥ - عبارات COM و OPEN COM

- أ - ما هو الغرض من عبارة OPEN COM ؟
- ب - ما هو الغرض من عبارة COM ؟
- ج - ما هو عدد أجزاء الاتصالات المدعمة بواسطة هذه العبارات؟
- د - ما هي العبارات الأخرى التي يجب أن تستخدم مع عبارة COM ؟

٩٦ - عبارة OPTION BASE

- أ - ما هما القيمتان المكنتان المستخدمتان مع عبارة OPTION BASE؟
- ب - هل تنقل اعدادات OPTION BASE إلى برنامج متسلسل؟
- ج - كيف يمكنك أن تحدد بطريقة أخرى حدود منظومة؟

٩٧ - عبارة PRINT

- أ - ما هو الاختلاف الذي يحدثه جزء STEP في عبارة PAINT؟
- ب - ماذا يعنى tiling؟

٩٨ - PALETTE و PALETTE USING

- أ - ماذا تعنى مجموعة الألوان PALETTE؟
- ب - ما مدى سرعة تغيير تأثير مجموعة الألوان على العرض الحالى؟
- ج - لماذا يستخدم جزء USING؟

٩٩ - عبارة PCOPY

- أ - ما هو عدد صفحات الشاشة فى حالة بطاقة CGA؟
- ب - متى تستخدم عبارة PCOPY؟

١٠٠ - دالة PEEK و عبارة POKE

- أ - ما هو القطاع الذى يتأثر بعبارة PEEK أو عبارة POKE؟
- ب - ما هو مدى جزء العنوان؟
- ج - ما هى القيمة التى تعيدها عبارة PEEK؟
- د - كيف يمكنك تغيير القطاع الحالى؟

١٠١ - دالة PEN

- أ - إلى أى الوحدات فقط تستخدم دالة PEN؟
- ب - ما هى العبارة التى يجب أن تنفذ قبل أن يمكن استخدام دالة PEN؟

١٠٢ - عبارات PEN ON و PEN OFF و PEN STOP

- أ - على أى الوحدات تستخدم عبارات اصطياد الأحداث هذه؟

١٠٣ - دالة وعبارة PLAY

- أ - ما هو الغرض من عبارة PLAY؟
- ب - ما هو الغرض من دالة PLAY؟
- ج - ما هو عدد النوت الموسيقية التي يمكن لعبها في الخلفية في نفس الوقت؟

١٠٤ - عبارات PLAY ON و PLAY OFF و PLAY STOP

- أ - ما هي الوحدات التي تتصيدها هذه العبارات؟
- ب - ماذا يحدث داخل البرنامج الفرعي لتشغيل حدث PLAY؟

١٠٥ - دالة PMAP

- أ - ما هو الغرض من دالة PMAP؟
- ب - لماذا تريد أن تعرف الاحداثيات المنطقية على الشاشة؟

١٠٦ - دالة POINT

- أ - كيف تستخدم دالة POINT في العودة مرة أخرى لنقطة أثناء رسم احدى الصور؟
- ب - ما هي القيمة التي تعيدها دالة POINT إذا كانت الاحداثيات خارج حدود الشاشة الحالية؟

١٠٧ - دالة POS

- أ - ما هي المعلومات التي تقدمها دالة POS؟
- ب - ما هي دلالة مؤشر العمود؟

١٠٨ - عبارة PRESET

- أ - ما هي دلالة جزء STEP في التكوين؟
- ب - ماذا يحدث للنقاط المرسومة عندما يحذف جزء اللون؟

١٠٩ - عبارة PRINT

- أ - أين تذهب المخرجات مع عبارة PRINT؟
- ب - إلى أي عدد من مناطق الطباعة تقسم الشاشة؟

ج - ما هو عرض مناطق الطباعة بالرموز؟

د - ما هي دلالة رموز التشكيل؟

١١٠ - عبارة PRINT USING

أ - ما هي دلالة الجزء USING؟

ب - ما هو أقصى عدد أرقام يمكنك أن تحدده لبيانات عددية؟

ج - ما هو رمز التشكيل للطباعة في الشكل الأسى؟

١١١ - عبارتا PRINT# و PRINT USING#

أ - ما هو الاختلاف بين هاتين العبارتين وعبارتي PRINT و PRINT USING؟

١١٢ - عبارة PSET

أ - ما هو الفرق بين هذه العبارة وعبارة PRESET؟

١١٣ - عبارة RANDOMIZE

أ - لماذا تستخدم دالة RANDOMIZE؟

ب - ما هي الطريقة المقنعة لضمان أن دالة RANDOMIZE تحصل على عدد فريد؟

١١٤ - عبارة READ

أ - ما هو الغرض من عبارة READ؟

ب - ما هي العبارة الأخرى المستخدمة مع عبارة READ؟

ج - ما هو الرمز المستخدم في فصل المتغيرات عن بعضها البعض في عبارة

READ؟

١١٥ - عبارة REDIM

أ - ما هو دليل المنظومة؟

ب - على أي نوع من المنظومات يمكنك استخدام عبارة REDIM؟

ج - متى تستخدم عبارة REDIM؟

١١٦ - عبارة REM

أ - ما هي الطريقة الأخرى لإدخال عبارات في برنامج بيسك السريع؟

ب - ما هو الغرض الآخر لاستخدام عبارة REM؟

١١٧ - عبارة RESET

أ - متى تستخدم عبارة RESET؟

١١٨ - عبارة RESTORE

أ - ما هو الغرض من عبارة RESTORE؟

ب - متى تحدد رقم سطر أو اسم سطر مع عبارة RESTORE؟

١١٩ - عبارة RESUME

أ - متى تستخدم RESUME 0؟

ب - متى تستخدم RESUME NEXT؟

ج - متى تستخدم RESUME line number line label؟

د - أين يمكن أن تحدث عبارة RESUME؟

١٢٠ - دالة RIGHT\$

أ - ماذا يحدث إذا كان جزء تعبير السلسلة فارغاً؟

ب - على أى جزء من السلسلة تعمل دالة RIGHT\$؟

١٢١ - دالة RND

أ - متى تستخدم دالة RND؟

ب - كيف تمنع دالة RND من اعادة نفس تسلسل الأعداد؟

ج - كيف يمكنك أن تحصل على أرقام صحيحة من دالة RND؟

١٢٢ - عبارة RUN

أ - ماذا يحدث لقيم المتغيرات عندما تستخدم عبارة RUN مع مواصفة سطر؟

ب - ماذا يحدث عندما ينفذ برنامج آخر باستخدام عبارة RUN؟

١٢٣ - دالة SADD

أ - ما هي القيمة التي تعيدها دالة SADD؟

ب - لماذا تستخدم دالة SADD؟

١٢٤ - دالة وعبارة SCREEN

- أ - ما نوع مطبع العرض الذى تدعمه عبارة SCREEN؟
- ب - ما هو جزء الصفحة النشطة فى عبارة SCREEN؟
- ج - ما هو جزء الصفحة المرئية فى عبارة SCREEN؟
- د - ما هى المعلومات التى تقدمها دالة SCREEN؟

١٢٥ - دالة وعبارة SEEK

- أ - ماذا تفعل دالة SEEK؟
- ب - ماذا تفعل عبارة SEEK؟
- ج - ما هى أقصى قيمة لجزء pos فى عبارة SEEK؟
- د - لماذا يجب أن تستخدم عبارة SEEK ودالة SEEK؟

١٢٦ - عبارة SELECT CASE

- أ - ما هو تكوين التحكم الذى يمكن أن توفره SELECT CASE بصورة أكثر ترتيباً؟
- ب - ما هو الرمز المستخدم فى فصل حالات الاختبار؟

١٢٧ - دالة SETMEM

- أ - ما هو الغرض من عبارة SETMEM؟
- ب - كيف يمكنك الحصول على حجم الكومة البعيدة مستخدماً دالة SETMEM؟

١٢٨ - دالة SGN

- أ - أين يمكنك استخدام دالة SGN؟

١٢٩ - عبارة SHARED

- أ - أين يمكن أن تظهر عبارة SHARED؟
- ب - ما هو الغرض من عبارة SHARED؟
- ج - كيف يمكن سرد متغيرات المنظومة فى عبارة SHARED؟

١٣٠ - عبارة SHELL

أ - ما هو الأمر الذى تكتبه للعودة إلى البرنامج بعد تنفيذ عبارة SHELL بدون مؤشرات؟

ب - ما هى المؤشرات التى تقبلها عبارة SHELL؟

١٣١ - دالة SIN

أ - ما هى وحدة القيمة التى تعيدها دالة SIN؟

ب - ما هو النوع التقليدى لنتيجة دالة SIN؟

١٣٢ - عبارة SOUND

أ - ما هو مدى جزء التردد فى عبارة SOUND؟

ب - كيف يمكنك اغلاق متحدث الكمبيوتر بعد عبارة SOUND؟

١٣٣ - دالة SPACE\$

أ - أين تستخدم دالة SPACE\$؟

ب - ما هى النوال التى يمكنها تحقيق نفس النتائج؟

١٣٤ - دالة SPC

أ - ما هو مدى القائمة لدالة SPC؟

ب - ما هى الدالة الأخرى التى يمكنها أن تحقق نتائج شبيهة؟

١٣٥ - دالة SQR

أ - فى أى التطبيقات يمكنك أن تستخدم دالة SQR؟

١٣٦ - عبارة STATIC

أ - ما هو الغرض من استخدام عبارة STATIC؟

ب - ماذا يحدث عندما توضح DEF FN و FUNCTION و SUB على أنها

؟STATIC

١٣٧ - أشباه الأوامر \$STATIC و \$DYNAMIC

- أ - ما هو نوع المنظومة الذى يوضحه شبه الأمر \$STATIC؟
- ب - ما هو نوع المنظومة الذى يوضحه شبه الأمر \$DYNAMIC؟
- ج - لماذا يجب أن تستخدم أشباه الأوامر هذه؟

١٣٨ - دالة STICK

- أ - عن أى وحدة تقدم دالة STICK معلومات؟

١٣٩ - عبارة STOP

- أ - ما هو الغرض من عبارة STOP؟
- ب - ماذا يحدث عندما تستخدم عبارة STOP فى برنامج قائم بذاته؟
- ج - ماذا يحدث عندما تستخدم عبارة STOP فى بيئة بيسك السريع؟

١٤٠ - دالة STR\$

- أ - لماذا تستخدم دالة STR\$؟
- ب - ما هو مكمل دالة STR\$؟

١٤١ - دالة STRIG

- أ - عن أى وحدة تقدم دالة STRIN معلومات؟
- ب - ما هو مدى جزء n فى التكوين؟

١٤٢ - عبارات STRING ON و STRING OFF و STRING STOP

- أ - أى الأحداث تتصيدها هذه العبارات؟
- ب - ما هو مدى جزء n فى التكوين؟

١٤٣ - دالة STRING\$

- أ - ما هى التطبيقات الممكنة لدالة STRING\$؟
- ب - ما هو أكبر حجم للسلسلة يمكنك أن تحدده فى دالة STRING\$؟

١٤٤ - عبارتا SUB و END SUB

- أ - هل يستطيع SUB أن يستدعى نفسه؟
- ب - لماذا تكتب برامج فرعية؟
- ج - كيف يمكنك إنتاج برنامج فرعى فى بيئة بيسك السريع؟

١٤٥ - عبارة SWAP

- أ - ماذا تفعل عبارة SWAP؟
- ب - ما هى التطبيقات الممكنة لعبارة SWAP؟
- ج - هل يمكن أن تكون قائمة العبارة من أى نوع؟

١٤٦ - عبارة SYSTEM

- أ - ماذا تفعل عبارة SYSTEM؟
- ب - هل يوجد لعبارة SYSTEM أى مؤشرات؟
- ج - ما هو الاختلاف بين تنفيذ برنامج قائم بذاته لعبارة SYSTEM وما إذا كان البرنامج ينفذ عندما ينفذ بيسك السريع عبارة SYSTEM؟

١٤٧ - دالة TAB

- أ - ما هو الغرض من دالة TAB؟
- ب - ماذا يحدث إذا كانت قائمة الدالة أكبر من عدد الأعمدة المتاحة؟

١٤٨ - دالة TAN

- أ - ما هى نتيجة دالة TAN؟
- ب - ما هو النوع التقليدى لنتيجة دالة TAN؟

١٤٩ - دالة وعبارة TIME\$

- أ - ما هو شكل وضع الوقت باستخدام عبارة TIME\$؟
- ب - ما هو شكل وضع الوقت الذى تعيده دالة STIME\$؟

١٥٠ - دالة TIMER

- أ - ما هى المعلومات التى تقدمها دالة TIMER؟
- ب - ما هى التطبيقات التى تناسبها دالة TIMER؟

١٥١ - عبارات TIMER ON و TIMER OFF و TIMER STOP

- أ - أى الأحداث تنصيدها هذه العبارات؟
- ب - ما هى العبارة الأخرى التى يجب أن تستخدم مع هذه العبارات؟

١٥٢ - عبارات TRON و TROFF

- أ - ما هو التتبع tracing؟
- ب - كيف يمكن استخدام عبارتى TRON و TROFF فى البرنامج؟
- ج - كيف يمكنك تتبع البرامج بطريقة أخرى؟

١٥٣ - عبارات TYPE و END..TYPE

- أ - ما هو الغرض من عبارتى TYPE و END..TYPE؟
- ب - ما هى العبارة الأخرى التى يجب استخدامها قبل أن يصبح نوع البيانات متاحاً فى البرنامج؟

١٥٤ - دالة VAL

- أ - ما هى الوظيفة التى تكملها دالة VAL؟
- ب - عند استخدام دالة VAL فإين يتوقف التحويل؟

١٥٥ - المتغيرات

- أ - ما هو المتغير؟
- ب - كيف توضح أحد المتغيرات فى بيسك السريع؟
- ج - ما هى خصائص المتغيرات فى بيسك السريع؟

١٥٦ - دالتا VARPTR و VARSEG

- أ - ما هو القرع؟
- ب - ما هو القطاع؟
- ج - أين يمكنك استخدام هاتين الدالتين؟

١٥٧ - دالة VARPTR\$

- أ - ما النتيجة التي تخرج من دالة VARPTR\$؟
- ب - أين يمكنك استخدام دالة VARPTR\$؟
- ج - ما هو نوع المنظومات المستخدمة كمؤشرات لهذه الدالة؟

١٥٨ - عبارة VIEW

- أ - ماذا تفعل عبارة VIEW؟
- ب - ما هو الغرض من جزء SCREEN في التكوين؟
- ج - ما هو مدى الرؤية view portal؟

١٥٩ - عبارة VIEW PRINT

- أ - ما هو الاختلاف بين عبارة VIEW وعبارة VIEW PRINT؟
- ب - كيف تؤثر العبارة على العرض المرئي؟

١٦٠ - عبارة WHILE.. WEND

- أ - ما هو نوع عبارة WHILE.. WEND؟
- ب - ما هي العبارات الأخرى التي تؤدي وظائف مماثلة؟

١٦١ - عبارة WIDTH

- أ - أي وحدة تؤثر عليها عبارة WIDTH؟
- ب - كيف تؤثر عبارة WIDTH بطرق أخرى على المخرجات؟
- ج - ما مدى سرعة اتخاذ عبارة WIDTH إجراءً؟

١٦٢ - عبارة WINDOW

- أ - ماذا تفعل عبارة WINDOW؟
- ب - ما هي معنوية جزء SCREEN في التكوين؟

١٦٣ - عبارتا WRITE و WRITE#

أ - ما هو الاختلاف بين عبارة WRITE وعبارة WRITE#؟

ب - ما هو الاختلاف بين عبارتي WRITE و WRITE# وعبارتي PRINT و PRINT#؟

قائمة بأهم المصطلحات

\$DYNAMIC,	شبيه أمر ديناميكي
\$INCLUDE, Metacommand,	شبيه أمر الشمول
\$STATIC,	شبيه أمر أستاذيكي
ABS,	دالة القيمة المطلقة
Array, Maximum size for an,	أقصى حد للمنظومة
Array for graphics GET and PUT,	حساب حجم منظومة لرسومات GET و PUT
Calculating size of,	
Array subscript, Lower limit of,	الحد السفلى لدليل المنظومة
Array subscript, Upper limit of,	الحد العلوى لدليل المنظومة
Arrays	منظومات
Changing the size of, 274	تغير الحجم
Declaring,	توضيح
Dynamic,	ديناميكية
Erasing,	حذف
Initializing,	وضع قيم ابتدائية
ASC,	دالة اسكى
Aspect,	وجه
ATN,	دالة قوس الظل
BASIC language implementation,	تنفيذ لغة البيسك
BEEP,	عبارة الصفير
Bits per pixel plane,	بت فى كل سطح نقاط رسم
BLOAD,	عبارة تحميل صورة الذاكرة
Branching, Conditional,	تفرع شرطى
Branching, Unconditional,	تفرع غير شرطى

BSAVE,	عبارة نسخ بيانات من الذاكرة
Buffer flushing,	تدفق الذاكرة الاحتياطية
BYVAL	كلمة تعرف كيف تمرر القائمة إلى البرنامج الفرعى
CALL,	عبارة استدعاء
CALL ABSOLUTE,	عبارة استدعاء مطلق
CALL INT86OLD,	عبارة تؤدي استدعاء DOS
CALL INT86OLD,	عبارة تؤدي استدعاء DOS
CALL INTERRUPT,	عبارة تؤدي استدعاء DOS
CALL INTERRUPTX,	عبارة تؤدي استدعاء DOS
CALLS,	عبارة استدعاء
Calling Non-Quick BASIC procedures,	استدعاء اجراءات غير بيسك السريع
Calling Quick BASIC procedures,	استدعاء اجراءات بيسك سريع
CDBL,	دالة لتحويل عددي إلى قيمة مزدوجة الدقة
CHAIN,	عبارة السلسلة
Characters, Non-printable,	رموز لا تطبع
CHDIR,	عبارة تغيير الدليل
CHR\$,	دالة الرمز
CINT,	دالة لتحويل تعبير عددي إلى قيمة صحيحة
CIRCLE,	عبارة الدائرة
CLEAR,	عبارة الاخلاء
CLNG,	دالة لتحويل تعبير عددي إلى قيمة صحيحة طويلة
CLOSE,	عبارة اغلق
CLS,	عبارة اخلاء الشاشة
COLOR,	عبارة اللون

Colors, Background,	ألوان الخلفية
Colors, Changing,	تغيير الألوان
COM,	قناة الاتصالات
COMMAND\$,	دالة الأوامر
Command-line compilation,	ترجمة سطر الأمر
Command-line options, QuickBASIC,	بدائل سطر الأمر
Command-line parameters,	مؤشرات سطر الأمر
Commands used with LIB.EXE,	أوامر مستخدمة مع LIB. EXE
Comments in a QuikBASIC program,	تعليقات في برنامج بيسك السريع
COMMON	عبارة مشاركة
COMMON Block,	مجموعة مشاركة
Communicating with QuickBASIC,	اتصالات ببيسك السريع
Compile options,	بدائل ترجمة
CONST,	ثابت
Constants, Types of,	انواع الثوابت
Control structures,	تكوينات التحكم
COS,	دالة جيب التمام
Counters,	عدادات
Creating a library,	إنتاج مكتبة
CSNG,	دالة لتحويل تعبير عددي إلى قيمة فردية الدقة
CSRLIN,	دالة تقدم موضع سطر نقطة البداية الحالي
Cursor control,	التحكم في نقطة البداية
Cursor movement commands,	أوامر حركة نقطة البداية
Cursor movement prefixes,	سابقات حركة نقطة البداية
Cursor scan lines,	أسطر فحص نقطة البداية
Cutting text,	قطع النص
CVD	دالة لتحويل قيمة سلسلة إلى قيمة عددية

CVDMBF,	دالة للتعامل مع شكل الثنائي لميكروسوفت
CVI,	دالة لتحويل قيمة سلسلة إلى قيمة عددية
CVL,	دالة لتحويل قيمة سلسلة إلى قيمة عددية
CVS,	دالة لتحويل قيمة سلسلة إلى قيمة عددية
CVSMBF,	دالة للتعامل مع الشكل الثنائي لميكروسوفت
DATA	بيانات
Data formatting,	تشكيل البيانات
Data Type conversion,	تحويل نوع البيانات
Data Types, QuickBASIC,	أنواع بيانات بيسك السريع
DATE\$,	دالة أو عبارة التاريخ
Date, Changing the system,	تغير تاريخ النظام
Date, Reading the system,	قراءة تاريخ النظام
DECLARE,	عبارة توضيح
DEF FN,	عبارة لتحديد دالة
DEF SEG,	عبارة لتحديد عنوان قطاع
DEFDBL,	عبارة لتعريف مجموعة حروف بأنها نوع بيانات بسيط
DEFINT,	عبارة لتعريف مجموعة حروف بأنها نوع بيانات بسيط
DEFLNG,	عبارة لتعريف مجموعة حروف بأنها نوع بيانات بسيط
DEFSNG,	عبارة لتعريف مجموعة حروف بأنها نوع بيانات بسيط
DEFSTR,	عبارة لتعريف مجموعة حروف بأنها نوع بيانات بسيط
Device drivers, Control of,	التحكم في مشغل الوحدة
Dialog boxes,	صناديق حوار

DIM,	عبارة البعد
Directory maintenance,	صيانة الدليل
DO,	عبارة أفعل
DOS, Exiting to,	الخروج إلى DOS
DOS, File handle,	معالجة ملفات DOS
DOS commands, Executing,	تنفيذ أوامر DOS
DOS environment variables, Reading,	قراءة متغيرات بيئة DOS
DOS environment variables, Setting,	إعداد متغيرات بيئة DOS
DOS Interrupts,	إنعاجات (اعتراضات) DOS
DOS shell,	رقية DOS
DRAW,	عبارة إرسم
DRAW Macro, Other commands,	أوامر أخرى لماكرو الرسم
Drawing figures,	رسم أشكال
Drawing lines,	رسم خطوط
EDITING, Multiwindow,	تنقيح متعدد النوافذ
Editor, QuickBASIC,	منقح بيسك السريع
ELSE,	عبارة والا
END,	عبارة النهاية
ENVIRON\$,	دالة البيئة
EOF,	دالة نهاية الملف
ERASE,	عبارة الحذف
ERDEV\$,	دالة حالة خطأ الوحدة
ERDEV,	عبارة حالة خطأ الوحدة
ERL,	دالة تحديد سطر الخطأ
ERR,	دالة الخطأ
ERROR,	عبارة الخطأ
Error codes, Device,	شفرات خطأ الوحدة

Error Codes, User-defined,	شفرات خطأ يعرفها المستخدم
Error message, QuickBASIC,	رسائل خطأ من بيسك السريع
Errors	أخطاء
Continuing after,	الاستمرار بعد
Mechanisms for trapping,	آلية الاصطياد
Events, Processing,	تشغيل احداث
Events, Trapping,	اصطياد احداث
EXE file, Creating an,	انتاج ملف تنفيذ
EXIT,	عبارة الخروج
EXP,	دالة الأس
Expression reordering in Quick BASIC,	إعادة أمر التعبير في بيسك السريع
Far Heap, Control of,	التحكم في الكومة البعيدة
FIELD,	عبارة الحقل
File access methods,	طرق الاتصال بالملف
File menu,	قائمة ملفات
File modes,	حالات الملف
File position,	موقع الملف
File processing,	تشغيل الملف
File size,	حجم الملف
FILEATTR,	دالة خواص الملف
Filenames, Generating,	انتاج ارقام ملفات
FILES,	عبارة ملفات
Files	ملفات
deleting,	حذف
Listing,	سرد
Renaming,	اعادة تسمية

Finding text,	إيجاد نص
FIX,	دالة اعادة الرقم الصحيح
Flow control, Program,	التحكم فى مسار البرنامج
FOR,	عبارة من
FRE,	دالة لتحديد الذاكرة المتاحة
FREEFILE,	دالة لتقديم رقم الملف المثالى المتاح
FUNCTION,	عبارة لتعريف دالة بيسك السريع
Function, Recursive,	دالة اعادة ذاتية
GET,	عبارة لقراء بيانات من ملف
GET, Graphics,	عبارة للتعامل مع الرسومات
GOSUB,	تكوين لاستدعاء مقطع فرعى
GOSUB, Nesting,	تكوين لتداخل استدعاء مقاطع فرعية
GOTO,	عبارة اذهب إلى
Help, Context sensitive,	مساعدة حساسه المحتوى
HEX\$,	دالة السادس عشرى
IF,	عبارة اذا
Immediate window,	نافذه فورية
Including files,	ملفات شمول
INKEY\$,	دالة لاعادة سلسلة من ب١٠ أو اثنين
INP,	عبارة لقراء بايت من بوابة الآلة
INPUT#,	عبارة مدخلات من ملف
INPUT\$,	عبارة مدخلات سلسلة من ملف
INPUT,	عبارة مدخلات من لوحة مفاتيح
Input, Data,	بيانات مدخلات
Input from a file, Data,	بيانات مدخلات من ملف
INSTR,	دالة تتعامل مع السلاسل
INT,	دالة لاعاده القيمة الصحيحة

IOCTL\$,	دالة للتداخل مع مشغل الوحدة
IOCTL,	دالة للتداخل مع مشغل الوحدة
Joystick control,	التحكم فى عصا اللعب
KEY,	عبارة لتحديد قيمة لمفاتيح الوظائف
KEYLIST,	عبارة لسرد تحديدات السلاسل لمفاتيح الوظائف على الشاشة
KEY(N) OFF,	عبارة لإلغاء اصطيايد الحدث للمفتاح N لحظيا
KEY (N), ON,	عبارة لتنشيط اصطيايد الحدث للمفتاح N
KEY(N) STOP,	عبارة لإيقاف اصطيايد الحدث للمفتاح N
Key sequences, User-defined,	تتابع مفاتيح يحدده المستخدم
Keyboard flags,	أشارات لوحة المفاتيح
Keyboard scan codes,	شفرات فحص لوحة المفاتيح
Keys, Input editing,	مفاتيح تنقيح المدخلات
Keys, Trapping extended,	مفاتيح توسيع الاصطيايد
KILL,	عبارة لإلغاء احد الملفات
Lables in QuickBASIC,	الاسماء فى بيسك السريع
LBQUND,	دالة الحد السفلى لدليل بعد المنظومة
LCASE\$,	دالة تحول قيمة سلسلة الى الحالة السفلية
LEPT\$	دالة لإعادة رموز من ناحية اليسار
LEN,	دالة لإعطاء طول السلسلة
LET,	دالة دع
Libraries, QuickBASIC,	مكتبات بيسك السريع
Lightpen control,	التحكم فى القلم الضوئى
LINE,	عبارة لرسم خط
LINE INPUT#,	عبارة لقبول مدخلات خط من ملف تتابعى
LINE INPUT,	عبارة لقبول مدخلات خط من لوحة المفاتيح
Line styling,	شكل الخط

Link options,	بدائل التوصيل
Linking,	توصيل
Loading a program,	تحميل برنامج
LOC,	دالة لتحديد موقع فى ملف
LOCATE,	عبارة لتحديد موقع نقطة البداية على الشاشة
LOCK,	عبارة اغلاق
LOF,	دالة لتقديم حجم الملف
LOG,	دالة اللوغاريتم الطبيعي
LOOP,	عبارة الدورة
LPOS,	دالة لتحديد موقع رأس الطابع
LPRINT,	دالة للطباعة بالطابع
LSET,	عبارة لنقل بيانات من الذاكرة إلى ذاكرة الملف الاحتياطية
LTRIM\$,	دالة لإزالة الفراغات
Memory	ذاكرة
Available,	متاحة
Loading multiple modules in,	تحميل مقاطع متعددة فى
Reading from,	قراءة من
Writing to,	كتابة فى
Memory address,	عنوان ذاكرة
Memory organization,	تنظيم الذاكرة
Metacommands,	اشباه أوامر
Metacommands, Writing,	كتابة أشباه الأوامر
Microsoft Binary Format,	شكل ثنائى من ميكروسوفت
MID\$,	دالة أو عبارة مع سلسلة مؤشر
MKD\$,	دالة تحويل تعبير مزدوج الدقة إلى قيمة سلسلة من 8 بايت

MKDIR,	عبارة لعمل دليل
MKDMBF\$,	دالة للتعامل مع أشكال ثنائية من ميكروسوفت
MKI\$,	دالة لتحويل تعبير صحيح إلى قيمة سلسلة من 2 بايت
MKL\$,	دالة لتحويل تعبير صحيح طويل إلى قيمة سلسلة من 4 بايت
MKS\$,	دالة لتحويل تعبير فردى الدقة إلى قيمة سلسلة من 4 بايت
MKSMBF\$,	دالة للتعامل مع أشكال ثنائية من مايكروسوفت
Music, Playing,	لعبة موسيقى
Music macro,	ماكرو موسيقى
NAME...AS,	دالة لإعادة تسمية الملفات
Nesting,	تتداخل
NEXT,	عبارة التالى
OCTS,	دالة لإعادة المكافئ الثنائى لتعبير ثمانى
ON,	عبارة عند
ON ERROR GOTO,	عبارة عند حدوث خطأ إذهب إلى
OPEN,	عبارة أفتح
OPTION BASE,	عبارة أساس البديل
OUT,	عبارة لإرسال بايت إلى بوابة الآلة
Output	مخرجات
Controlling,	التحكم فى
Formatting,	تشكيل
Writing,	كتابة
PAINT,	عبارة الدهان
PALETTE,	عبارة تغيير الألوان
Parameter passing,	مرور المؤشر

Parse,	قراءة من عناصر مميزة
Pasting text,	لصق النص
PCOPY,	عبارة نسخ الشاشة
PEEK,	دالة لتحديد موقع ذاكرة
PEN,	دالة لإحداثيات القلم الضوئي
PEN OFF,	دالة لإلغاء اصطياذ نشاط القلم الضوئي
PEN ON,	دالة لتنشيط اصطياذ نشاط القلم الضوئي
PEN STOP,	دالة لإيقاف اصطياذ نشاط القلم الضوئي
Pi, Value of,	قيمة ط (Π)
Planes,	مستويات
PLAY,	دالة وعبارة لعب الموسيقى
PLAY OFF,	عبارة لإلغاء اصطياذ الحدث
PLAY ON,	عبارة لتنشيط اصطياذ الحدث
PLAY STOP,	عبارة لإيقاف اصطياذ الحدث
Plotting points,	نقاط رسم
PMAP,	دالة للتعامل مع الأحداثيات
POINT,	دالة لأحضار رقم أو أحداثيات نقاط الرسم
POKE,	دالة لكتابة بيانات في موقع ذاكرة
POS,	دالة لتحديد عمود نقطة البداية
PRINT#,	عبارة كتابة في ملف
PRINT,	عبارة كتابة على الشاشة
PRINT, USING,	عبارة الكتابة بأستخدام شكل معين
Printing data,	طباعة بيانات
PSET,	عبارة لرسم نقطة على الشاشة
PUT,	عبارة لكتابة بيانات في ملف
PUT, Graphics,	عبارة للتعامل مع رسومات
QuickBASIC	بيسك السريع

Exiting,	خروج
Installing,	تشبيد
Running,	تشغيل
QuickBASIC disks, contents,	محتويات أقراص بيسك السريع
QuickBASIC environment,	بيئة بيسك السريع
Random number generator, Reseed-	إعادة وضع قيمة للأساس لمنتج أرقام عشوائية -
ing,	
Random numbers, Generating,	إنتاج أرقام عشوائية
RANDOMIZE,	عبارة لبدأ إنتاج أرقام عشوائية
READ,	عبارة قراءة
REDIM,	عبارة للتعامل مع منظومات
REM,	عبارة ملحوظات
Replacing text,	إحلال نص
RESET,	عبارة لإغلاق كل ملفات القرص
RESTORE,	عبارة لاستعادة برامج بيسك
RESUME,	عبارة لاستمرار تنفيذ البرنامج
RETURN,	عبارة للعودة من ملف فرعى
RIGHT\$,	دالة لإعادة عدد رموز من ناحية اليمين
RMDIR,	عبارة لإزالة دليل
RND,	دالة تعيد رقم عشوائى
RSET,	عبارة لنقل بيانات من الذاكرة إلى ذاكرة الملف الاحتياطية
RTRIM\$,	دالة لإزالة الفراغات
RUN,	عبارة تشغيل
Running LIB.EXE,	تشغيل ملف LIB. EXE
SADD,	دالة لتقديم عنوان تعبير سلسلة
Saving a program,	حفظ البرنامج

SCREEN,	دالة وعبرة الشاشة
Screen coordinates,	إحداثيات الشاشة
Screen coordinates, Relative,	إحداثيات الشاشة النسبية
Screen modes,	حالات الشاشة
Screen pages,	صفحات الشاشة
SEEK,	دالة وعبرة البحث
SEG,	كلمة تعرف كيف تمرر القائمة إلى البرنامج الفرعى
SELECT CASE,	عبرة اختيار الحالة
SETMEM,	عبرة للتعامل مع الكومة البعيدة
SGN,	دالة الإشارة
SHARED,	عبرة مشاركة
SHELL,	عبرة خروج من البرنامج وتنفيذ أمر DOS والعودة
SIN,	دالة الجيب
SOUND,	دالة الصوت
SPACE\$,	دالة الفراغات
SPC,	دالة القفز
SQR,	دالة الجذر التربيعى
Stack, Modifying the size of,	تعديل حجم الرصة
STATIC,	عبرة للتعامل مع المتغيرات والمنظومات
STICK,	دالة لقراءة إحداثيات عصا اللعب
STOP,	دالة لإيقاف تنفيذ البرنامج
STR\$,	دالة تحويل تعبير عددى إلى سلسلة
STRIG,	دالة لتعيد حالة إطلاق عصا اللعب
STRIG OFF,	عبرة لإلغاء تنشيط اصطلياد حدث على عصا اللعب

STRIG ON,	عبارة تنشيط اصطيات حدث على عصا اللعب
STRIG STOP,	عبارة إيقاف تنشيط اصطيات حدث على عصا اللعب
STRING\$,	دالة تعيد سلسلة لأحد رموز اسكى
String, Case conversion of,	تغيير حالة السلسلة
String storage compaction,	ضغط تخزين السلسلة
Stringd, Manipulating,	معالجة السلاسل
Strings, Parsing,	قراءة من سلاسل مميزة
SUB..END SUB,	عبارتان تعرفان برنامج فرعى
Subprograms,	برامج فرعية
Substrings exeuction,	تنفيذ سلسلة فرعية
SWAP,	عبارة لتبادل متغيرات مؤشر
SYSTEM,	عبارة لاجلاق كل الملفات والعودة إلى نظام التشغيل
TAB,	دالة لترحيل مخرجات البرنامج للداخل
TAN,	دالة الظل
Terminating Program,	برنامج إنهاء
THEN,	بعد ذلك
Tiling,	طلاء
TIME\$,	دالة وعبارة الوقت
Time, Reading the system,	قراءة وقت النظام
Time, Setting the system,	وضع وقت النظام
TIMER,	دالة تعطى الوقت المنقضى بعد الساعة ١٢ بعد الظهر
TIMER OFF,	عبارة لإلغاء تنشيط اصطيات الحدث لدالة TIMER
TIMER ON,	عبارة تنشيط اصطيات الحدث لدالة TIMER

TIMER STOP,	عبارة لإيقاف اصطيات الحدث لدالة TIMER
Tracing, Program execution,	تتبع تنفيذ البرنامج
TROFF,	عبارة توقف تنشيط اصطيات حدث تتبع تنفيذ البرنامج
TRON,	عبارة تنشيط تتبع تنفيذ البرنامج
Truncating numeric data,	إلغاء بيانات عددية
Type declaration, Mass,	توضيح الملف
TYPE...END TYPE,	عبارة لوصف متغيرات يعرفها المستفيد
Typed files,	ملفات مكتوبة
Typing a program in QuickBASIC,	كتابة برنامج بيسك السريع
UBOUND,	دالة لتحديد الحد العلوى لدليل بعد المنظومة
UCASE\$,	دالة تحول قيمة السلسلة إلى الحالة العليا
UNLOCK,	عبارة عدم الإغلاق
VAL,	دالة لتحويل تعبير سلسلة إلى قيمة عددية
Variable, Memory address of a,	عنوان ذاكرة
Variable creation,	إنتاج متغير
Variable movement in quickBASIC,	حركة متغير فى بيسك السريع
Variable scope,	مدى متغير
Variable scope, Controlling,	التحكم فى مدى متغير
Variable storage in Quick BASIC,	تخزين متغير فى بيسك السريع
Variables,	متغيرات
Assigning values to,	تحديد قيمة لـ
Declaring,	توضيح
Size of,	حجم
User-defined,	يعرفه المستفيد
VARPTR\$,	دالة تقدم تمثيل سلسلة لقطاع وفرع متفرع
	مؤشر

VARPTR,
VARSEG,
VIEW,
VIEW PRINT,
WAIT,
WHILE.. WEND,
WIDTH,
WIDTH LPRINT,
WINDOW,
WRITE#,
WRITE,
Writing subroutines,

دالة للحصول على قطاع وفرع الذاكرة لمتغير
دالة للحصول على قطاع وفرع الذاكرة لمتغير
عبارة تعرف حدود الرسومات للشاشة
عبارة تعرف حدود النصوص للشاشة
عبارة إنتظار
عبارة لتكرار تنفيذ مجموعة عبارات
عبارة عرض السطر
عبارة عرض سطر الطابع
عبارة النافذة
عبارة الكتابة في ملف
عبارة الكتابة على الشاشة
كتابة مقاطع فرعية

رقم الإيداع

٩٣ / ٨٦٩٣

 **مطابع العصر الحديث**
MODERN EGYPTIAN PRESS
ت : ٢٢١١٠٧١ - ٢٢١١٠٧٢ - فاكس ٢٢١١٠٧٣

هذا الكتاب

لقد ظهرت لغة البيسك منذ خمسة وعشرين عاماً كلغة بسيطة لتعليم المبتدئين أساسيات البرمجة باستخدام الكمبيوتر. وشهدت هذه اللغة البسيطة تطورات هائلة على مدار هذه السنين جعلتها في مصاف لغات البرمجة الأساسية حالياً. ويرجع الفضل في جعل هذه اللغة قادرة على دعم البرمجة

المرتبة إلى صيغ كويك بيسك أو بيسك السريع. وفيما يلي بعض مميزات صيغ بيسك السريع عن صيغ بيسك السابقة:

- إمكانية كتابة الأسطر دون وضع أرقام لها.
 - استخدام نوع جديد من أنواع المتغيرات العددية وهو الرقم الصحيح الطويل long integer.
 - التوسع في مكونات التحكم التي سبق استخدامها في صيغ البيسك السابقة لها بإدخال مجموعات IF ومجموعات SELECT CASE ودورات DO.
 - استخدام البرامج الفرعية والتي تدعم البرمجة المربطة.
 - استخدام الملفات الثنائية، واستخدام مترجمات compilers للغة لترجمة البرامج.
 - استخدام المقاطع التي يمكن معالجتها كأجزاء مستقلة، واستخدام خاصية مشاركة المتغيرات عبر المقاطع المختلفة.
- وترجع أهمية هذا الكتاب لحداثة المادة العلمية المقدمة فيه وكذلك لسهولة عرضها وترتيب مواضعها: وعلى الرغم من أن الكتاب مقدم للنارئ المبتدئ وكذلك للمبرمجين إلا أنه لا يحتاج لشيء في دراسته سوى الألمام باستخدام لوحة مفاتيح الكمبيوتر ونظام تشغيله. وهذه متطلبات ميسورة لمن يرغب في تعلم هذه اللغة المهمة.

والله الموافق

الناشر

علاؤ الدين خليل

ISBN : 977- 5201- 46- 2

ACADEMIC BOOKSHOP

